Universidad de las Ciencias Informáticas

FACULTAD 5

ENTORNOS VIRTUALES

Título: Biblioteca para la manipulación de videos digitales en Sistemas de Realidad Virtual

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Orlay García Ducongé.

Tutor: Ing. Fernando Jiménez López.

Co-tutor: Ing. Yanoski R. Camacho Román.

Ciudad de la Habana, Junio del 2007

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Orlay García Ducongé

_____

Firma del Autor

Ing. Fernando Jiménez López

_____

Firma del Tutor

DATOS DE CONTACTO

Nombre y Apellidos: Fernando Jiménez López

Edad: 28 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Informática

Categoría Docente: Profesor Instructor

E-mail: fjimenez@uci.cu

Graduado en Ingeniería Informática, en la Ciudad Universitaria José Antonio Echevarría (CUJAE), con cuatro años de experiencia en el tema de la Gráfica Computacional. Actualmente Profesor Instructor de la Universidad de la Ciencias Informáticas impartiendo la asignatura de Gráficos por Computadora, además de ser el jefe del Polo de Realidad Virtual en la Facultad 5 de la Universidad de las Ciencias Informáticas.

AGRADECIMIENTOS

No soy el único artífice de que llegara el momento de escribir estas líneas, son muchas las personas a las que les agradezco por el apoyo y ayuda que me han dado durante estos cinco años de carrera, a todos ellos muchas gracias.

Un especial agradecimiento:

A la Universidad de las Ciencias Informática.
A mis padres, por darme la oportunidad de hacer realidad mis sueños.
A mi hermana, mis abuelas y demás familiares que tanto me quieren.
A mi novia, a quien quiero tanto.
A todos mis compañeros de cuarto que soportaron la convivencia conmigo, durante 5 años. (Especialmente a Yessy, Yasmany, Abdelaziz, Leonardo, Yordanis).
A todos mis profesores.
A mi tutor Fernando y a su esposa Leticia por la ayuda brindada.
Y a todos mis amigos que en algún momento de mi vida me han ayudado o han compartido parte de su tiempo conmigo.
Sin todos ellos habría sido imposible llegar hasta aquí.

Muchas gracias.

DEDICATORIA

…este trabajo de diploma está dedicado a mi familia y amigos, a ellos les debo lo que soy.

Orlay.

RESUMEN

Este trabajo propone una biblioteca para el manejo de videos digitales. Surge a partir de la necesidad de incorporarle a la herramienta SceneToolkit funcionalidades de manipulación de videos digitales.

El trabajo se centra en los sistemas para manejo de videos digitales en Sistemas de Realidad Virtual. Se propone como variante de solución el empleo de funciones de la API de Windows, sobre la plataforma Windows, y de la biblioteca avifile-0.7, sobre la plataforma Linux, para leer los ficheros de videos digitales y reproducir el audio. Además se propone el uso de la biblioteca gráfica OpenGl para la visualización de los videos en un entorno virtual.

El presente trabajo se propone obtener una biblioteca que brinda las funcionalidades básicas para la manipulación de videos digitales, que pueda ser utilizada por todo aquel que pretenda incorporarle un video de extensión (.avi) a su aplicación gráfica.

PALABRAS CLAVE

Video
Video Digital
Manipulación de Videos
OpenGl
Texturizar
Realidad Virtual
Entornos Virtuales

SUMMARY

This paper proposes a library for managing digital videos. It arises from the need to incorporate the tool SceneToolkit features of digital video manipulation.

The work focuses on systems for managing digital video in Virtual Reality Systems. It is proposed as an alternative solution using functions of the Windows API on the Windows platform, and the avifile-0.7 library on the Linux platform, to read the files of digital video and audio playback. Besides, proposing the use of the OpenGL graphics library for viewing videos in a virtual environment.

This paper intends to obtain a library that offers basic functionality for handling digital video, which can be used by anyone who seeks to incorporate a video extension (. avi) to its application graphically.

TABLA DE CONTENIDOS

INTRODUCCIÓN

A pesar de haber surgido hace ya algunos años, el fenómeno denominado Realidad Virtual (RV) continúa revolucionando el mundo, no sólo de la informática sino también de diversas áreas como la medicina, la arquitectura, la educación…

A finales de los 80 surgió la necesidad de crear un espacio totalmente interactivo, generado a través de la tecnología, por lo que los gráficos por computadoras entraron en una nueva época. Se comenzaron a remplazar los enfoques bidimensionales y de dibujo de líneas (2D) por las soluciones tridimensionales (3D), las cuales se han visto enriquecidas con sensaciones del mundo real a través de estímulos visuales, auditivos y de otros tipos que afectan al usuario de manera interactiva.

Con el surgimiento de los entornos virtuales, los mundos que se representan pueden no existir (ser producto de la imaginación humana), o una copia exacta del mundo real. Un entorno virtual es una forma de representar la realidad o la imaginación, pero de forma digital.

En el desarrollo de productos relacionados con la Realidad Virtual se destacan los juegos y simuladores. En el caso de los juegos lo importante es la modalidad (cómo se tiene que jugar), pero sin duda la historia del juego gusta mucho a los jugadores. La tendencia más marcada en la actualidad para narrar estas historias es a través de videos. Por su parte, en los simuladores los videos se emplean sobre todo en las presentaciones, y en ocasiones se utilizan para dar instrucciones en un momento determinado. La utilización, cada vez más frecuente, de videos en entornos virtuales ha implicado que los usuarios de estas aplicaciones exijan cada vez más la presencia y calidad de los mismos.

En Cuba ya se ha comenzado a explorar el mundo de la Realidad Virtual. Actualmente en la Universidad de las Ciencias Informáticas se trabaja en el desarrollo de una herramienta básica para el manejo gráfico, que ya cuenta con diferentes subsistemas, pero carece de funcionalidades para la manipulación de videos digitales.

Varios proyectos de la Facultad 5 que se encuentran vinculados a la Realidad Virtual, utilizan esta herramienta SceneToolKit para desarrollar sus productos. Los productos que se obtienen en estos proyectos se ven afectados en cuanto a la posibilidad de poder manipular videos digitales, sin duda un

importante recurso dentro del mundo de los entornos virtuales. Esto imposibilita la utilización de la herramienta en la creación de video-juegos y simuladores que puedan competir en el mercado actual.

Analizando detenidamente la problemática anterior, este trabajo propone como problema científico a resolver: ¿Cómo lograr funcionalidades de manipulación de videos digitales dentro de los entornos de Realidad Virtual, integrables a la herramienta SceneToolKit?

De esta forma se plantea como objeto de investigación, los sistemas para manejo de videos digitales. El trabajo se basará en el manejo de videos digitales dentro de los Sistemas de Realidad Virtual.

Se tendrá como Objetivo General de Investigación:

- Desarrollar una biblioteca para la manipulación de videos digitales que se pueda integrar a la herramienta SceneToolKit.

Para lo cual se proponen las siguientes Tareas de Investigación:

- Investigar los diferentes formatos de videos digitales que existen.
- Seleccionar un formato de video digital con el que se trabajará.
- Analizar las técnicas ya existentes para la manipulación de videos digitales, dependiendo del formato seleccionado, en Sistemas de Realidad Virtual.
- Estudiar la arquitectura e interfaz de la herramienta SceneToolKit.
- Diseñar una biblioteca para la manipulación de videos digitales con una arquitectura compatible con la herramienta SceneToolKit.
- Implementar una biblioteca para la manipulación de videos digitales.

En sentido general con este trabajo se espera poder utilizar de forma sencilla videos en las aplicaciones gráficas que se desarrollen utilizando la herramienta SceneToolKit.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se explicarán los conceptos fundamentales relacionados con el video digital, así como algunas de las características de los formatos de video más comunes en el mundo de la computación. Se abordará más profundamente sobre las características del formato AVI. Además se mencionan las características de las bibliotecas utilizadas para leer los ficheros de video.

1.1 ¿Qué es el video?

Etimológicamente la palabra video proviene del verbo latino videre, y significa "yo veo". [6]

El video no es nada más que la reproducción en forma secuencial de imágenes, que al verse con una determinada velocidad y continuidad dan la sensación al ojo humano de apreciar el movimiento natural. Además de la imagen, el otro componente esencial es el sonido. [4]

Podemos definir un video como la mezcla en un único fichero de un conjunto de sonidos e imágenes que conjuntamente transmiten un mensaje al usuario.

En el aspecto gráfico, un video se compone de una secuencia de imágenes denominadas fotogramas (también frames o cuadros), cada una de las cuales aparece en pantalla un determinado espacio de tiempo, suficiente para crear en el espectador la sensación de continuidad entre fotogramas, generando así la visión global de una única escena en movimiento.

Las imágenes pueden ser sintetizadas (creadas manualmente) o captadas a partir del entorno (video). Al igual que en el caso de las imágenes estáticas, los ficheros pueden ser muy voluminosos, y tienen unas capacidades de modificación limitadas. [3]

1.2 Historia del video.

La innovación en el registro de imágenes visuales y auditivas en formato de video, comenzó a finales de los años cincuenta e inicios de los sesenta, es entre 1965 y 1978 cuando se consolida como un medio con singularidad y aplicaciones propias.

Una fecha importante en la historia de este medio es 1964, durante los juegos olímpicos de Tokio: primer acontecimiento donde se hace una reprogramación diferida de la transmisión en directo. En 1965 se efectúa el primer video personal con una intención artística, considerándose este año como el nacimiento del “videoarte”.

Posteriormente, en 1968, la Sony Corporation produce el “portpack” primera cámara portátil comercializada, y en ese mismo año acontece que Jean Louis Godard graba la revuelta francesa de estudiantes, -hecho conocido con el Mayo Francés-, material que era visto por la noche en una librería francesa, naciendo así el video-reportaje y el video documental.

A partir de 1970 Philips lanza el sistema VCR y otorga facilidades de utilización al ciudadano común, que encuentra nuevas posibilidades de uso al registrar experiencias cotidianas, familiares y sociales, y amplía sus posibilidades expresivas, renueva la forma de transmitir información, delata injusticias, implementa la vigilancia de bancos y lugares públicos, apoya la investigación, entre otros muchos usos. [5]

En EEUU, en 1979, se producen las primeras aplicaciones comerciales de video interactivo. Y en 1980, sale al mercado el primer reproductor láser de tipo doméstico. A principios de los años 80 se inició el desarrollo de equipos para almacenar información en formato óptico, este tipo de tecnología supuso la posibilidad de almacenar una mayor información en un espacio menor, y por lo tanto un paso imprescindible para el almacenamiento de imágenes en soporte informático. Al soporte desarrollado se denominó videodisco y aportaba una importante característica para el desarrollo posterior de las multimedias, y es que su lector era fácilmente controlable por medio de un ordenador. [7]

1.3 Digitalización.

Cuando se comienza a digitalizar un video hay que tener presente que es necesario digitalizar la información referente a la imagen y la información referente al audio.

Cuando la información a digitalizar es la de las imágenes, cada cuadro de la imagen es muestreado en unidades de pixeles, con lo que los datos a almacenar serán los correspondientes al color de cada pixel.

Para digitalizar una señal de video analógico es necesario muestrear todas las líneas de video activo. La información de brillo y color son tratadas de forma diferente por el sistema visual humano, ya que es más sensible al brillo que al color. Con lo que se usa un componente especial para representar la información del brillo, la luminancia, una para el color y la saturación, la crominancia. Cada muestra de color se codifica en señal Y-U-V (Y- luminancia, U y V crominancia) partiendo de los valores del sistema RGB. Con este sistema las diferencias de color pueden ser muestreadas sin resultados visibles, lo que permite que la misma información sea codificada con menos ancho de banda.

Las imágenes de video están compuestas de información en el dominio del espacio y el tiempo. La información en el dominio del espacio es provista por los pixels, y la información en el dominio del

tiempo es provista por imágenes que cambian en el tiempo. Puesto que los cambios entre cuadros colindantes son diminutos, los objetos aparentan moverse suavemente.

El valor de luminancia de cada pixel es cuantificado con ocho bits para el caso de imágenes blanco y negro. En el caso de imágenes de color, cada pixel mantiene la información de color asociada; una imagen completa es una composición de tres fotogramas, uno para cada componente de color, así los tres elementos de la información de luminancia son cuantificados a ocho bits.[4]

Por otra parte, cuando la información a digitalizar es la del audio, consiste básicamente en realizar de forma periódica muestras de la señal. Según la periodicidad de muestras que realicemos la conversión de analógico a digital será de mayor o menor calidad.

En la conversión analógica-digital del audio el muestreo es uno de los procesos que permite la digitalización de las señales. Estas muestras no se toman de forma aleatoria (al azar), sino que se toman intervalos fijos de tiempo.

Cada muestra debe durar el mismo tiempo y efectuarse en el mismo intervalo. La velocidad a la que se hace este muestreo, es decir, el número de muestras que se toman por segundo, es lo que se conoce como frecuencia de muestreo.

Por muy eficaz que sea el muestreo realizado y por muy alta que sea la frecuencia de este, hay que tener presente que siempre que haya un muestreo va a haber una cierta pérdida de calidad de la señal. Siempre habrá matices de la señal que no van a ser tenidos en cuenta, dado que no han sido muestreados, pero si realizamos un muestreo de buena calidad, luego podremos reconstruir la señal de manera prácticamente idéntica a la original.

Luego es necesario realizar un proceso de filtrado que consiste en la realización interna de un procesamiento de datos de entrada. El valor de la muestra de la entrada actual y algunas muestras anteriores (que previamente habían sido almacenadas) son multiplicados por unos coeficientes definidos. Finalmente los resultados de todas estas multiplicaciones son sumados, dando una salida para el instante actual.

Los filtros digitales se usan frecuentemente para atenuar las componentes de baja frecuencia, las componentes de alta frecuencia y dejar pasar un determinado rango de frecuencias de una señal y atenuar el paso del resto. [18]

1.4 Formatos de video digital.

La reproducción de video es ya un elemento común en los ordenadores modernos, siendo esta una de las tareas que más recursos consume. La sensación de movimiento se consigue con secuencias de imágenes con una velocidad de unos 30 fotogramas por segundo. Si el objetivo es video a través de Internet, las limitaciones del ancho de banda de la red presentan muchos problemas. Las posibilidades se reducen de nuevo al empleo de sistemas de compresión y a la reducción del tamaño de las ventanas de video y del número de fotogramas por segundo.

Una fotografía sin comprimir de 800x600 píxeles de resolución ocupa aproximadamente 1,3 Mb. Así pues, secuencias de video de este tamaño y con 30 fotogramas por segundo generarían ficheros de video con un tamaño de 390 Mb para 10 segundos ó 11,4 Gigabytes para 5 minutos. Por tanto, son imprescindibles drásticos sistemas de compresión para el manejo de video. [17]

Los estándares de video digital más conocidos son: MPEG, Quicktime, MOV, AVI, real video, ASF,...
Y para video analógico: NTSC, PAL, SECAM.

1.4.1 Formato Avanzado de Secuencias.

ASF fue desarrollado por Microsoft en 1996. Es uno de los primeros formatos de ficheros designados específicamente para el streaming. Este formato está optimizado para enviar secuencias multimedia a través de una red y es el recomendado para ello pues tiene la capacidad de adaptarse a variables anchos de banda y cambios en las condiciones de la red. Es un estándar abierto que admite la entrega de datos a través de una gran variedad de protocolos y redes.

También es posible utilizar cualquier códec para codificar las secuencias ASF. Se utiliza para ordenar, organizar y sincronizar los datos multimedia que se transmitirán por las redes. Sin embargo, puede utilizarse para especificar el formato de las presentaciones en directo y es también adecuado para la reproducción local.

Es un formato de alta flexibilidad que contiene una descripción y una representación digital comprimida de audio, video, imágenes, subtítulos y eventos. [10]

1.4.2 Real Video.

Real Video es un formato propietario de video desarrollado por RealNetworks. Fue liberado en 1997 y a partir de 2006 se encuentra en la versión 10. Real Video es compatible con muchas plataformas, incluyendo Windows, Mac, Linux, Solaris y varios teléfonos móviles.

Suele ser emparejado con Real Audio y empaquetado en Real Media (.rm). Real Media es adecuado para su uso como formato streaming, es uno de los formatos que se tiene en cuenta para enviar video a través de la red. Se puede utilizar para ver televisión en directo, ya que no requiere descargar el video de antemano. [19]

Es un flujo de datos contínuo que permite a un archivo estándar de video, tal como puede ser MPEG, ser visualizado a través de Internet. Esto quita la necesidad de transmitir el archivo entero antes de visualizarlo. [10]

1.4.3 Windows Media Video.

El formato .wmv (Windows Media Video) es una extensión que no tiene diferencia con los archivos .asf. Estos usan el formato de fichero estándar de Windows Media. Los ficheros con extensión .asf normalmente son utilizados en contenidos basados en Windows Media usando las herramientas Windows Media 4.0. Los ficheros .wma (Windows Media Audio) y .wmv se introdujeron como un convenio con la versión 7 para posibilitar a los usuarios la fácil diferenciación entre los ficheros de audio .wma y los de video .wmv. Sin embargo, no hay ninguna limitación en el formato, y las extensiones pueden usarse intercambiablemente.

Algunas herramientas y servicios que fueron creados para ser usados con las versiones anteriores de las Tecnologías Windows Media requieren la extensión de .asf en orden para aceptar el contenido. Se puede simplemente renombrar cualquier archivo .wma o .wmv para usar la extensión de .asf y usarlos con esas herramientas. [10]

1.4.4 Formato MPEG.

MPEG (Grupo de Expertos en Imágenes en movimiento) es un estándar internacional, definido por un comité llamado MPEG formado por la ISO, para la representación codificada y comprimida de imágenes en movimiento y audio asociado, orientado a medios de almacenamiento digital.

El algoritmo que utiliza además de comprimir imágenes estáticas compara los fotogramas presentes con los anteriores y los futuros para almacenar sólo las partes que cambian. La señal incluye sonido en calidad digital. El inconveniente de este sistema es que debido a su alta complejidad necesita apoyarse en hardware específico.

MPEG aplica la compresión temporal y la espacial. Los bloques de imagen y los de predicción de errores tienen una gran redundancia espacial, que se reduce gracias a la transformación de los bloques desde el dominio del espacio al dominio de frecuencia. MPEG requiere una intensiva computación para su codificación.

Existen diferentes opciones dependiendo del uso:

MPEG-1

Guarda una imagen, la compara con la siguiente y almacena sólo las diferencias. Se alcanzan así grados de compresión muy elevados.

Consigue el mayor grado de compresión a costa de un mayor tiempo de cálculo. Es un estándar escogido por Video-CD: calidad VHS con sonido digital.

MPEG-2

Con una calidad superior al MPEG-1, fue universalmente aceptado para transmitir video digital comprimido con velocidades mayores de 1Mb/s aproximadamente. Pueden conseguirse elevados ratios de hasta 100:1, dependiendo de las características del propio video.

Normalmente define dos sistemas de capas, el flujo de programa y el flujo de transporte. Se usa uno u otro pero no los dos a la vez. El flujo de programa funcionalmente es similar al sistema MPEG-1. La técnica de encapsulamiento y multiplexación de la capa de compresión produce paquetes grandes y de varios tamaños. Los paquetes grandes producen errores aislados e incrementan los requerimientos de buffering en el receptor/decodificador para demultiplexar los flujos de bits. En contraposición el flujo de transporte consiste en paquetes fijos de 188 bytes lo que decrementa el nivel de errores ocultos y los requerimientos del receptor.

MPEG4

Es un estándar relativamente nuevo orientado inicialmente a las videoconferencias, y para Internet. El

objetivo es crear un contexto audiovisual en el cual existen unas primitivas llamadas AVO (objetos audiovisuales). Se definen métodos para codificar estas primitivas que podrían clasificarse en texto y gráficos.

Ha sido especialmente diseñado para distribuir videos con elevados ratios de compresión, sobre redes con bajo ancho de banda manteniendo una excelente calidad para usuarios con buen ancho de banda. Es rápido codificando el video de alta calidad, para contenidos en tiempo real y bajo demanda. [4]

1.4.5 Quicktime.

El formato Quicktime (ficheros MOV) fue creado por Apple para el uso en computadoras Macintosh, aunque se ha extendido a otras plataformas. Es un software que le permite reproducir y editar video digital, así como otros tipos de archivos, en el ordenador. Quicktime no es en sí ninguna aplicación, sino una tecnología que permite a las aplicaciones llevar a cabo diversas funciones. Consta de una serie de elementos de software que amplían la capacidad del sistema operativo para gestionar archivos dinámicos. [10]

1.4.6 MJPEG.

Motion-JPEG es una versión extendida del algoritmo JPEG que comprime imágenes. Básicamente consiste en tratar al video como una secuencia de imágenes estáticas independientes a las que se aplica el proceso de compresión del algoritmo JPEG una y otra vez para cada imagen de la secuencia de video. Existen cuatro modos de operación para el JPEG: secuencial, progresiva, sin pérdida, y jerárquica. Normalmente se utiliza el modo secuencial.

La ventaja es que se puede realizar en tiempo real e incluso con poca inversión en hardware. El inconveniente de este sistema es que no se puede considerar como un estándar de video pues ni siquiera incluye la señal de audio. Otro problema es que el índice de compresión no es muy grande.

Motion-JPEG es el método elegido para las aplicaciones donde se envía la misma información a todos los usuarios, los broadcast. [4]

1.4.7 AVI.

Es el acrónimo de Audio Video Interleave (intercalado de audio y video). Se trata de un formato de archivo que actúa como contenedor de flujos de datos de audio y video. [9]

"Intercalado" significa que en un fichero AVI los datos de audio y video son almacenados consecutivamente en capas (un segmento de datos de video es seguido inmediatamente por otro de audio). Es el formato más extendido para el manejo de datos de audio/video en una PC. [12]

El formato avi permite almacenar simultáneamente un flujo de datos de video y varios flujos de audio. El formato concreto de estos flujos no es objeto del formato AVI y es interpretado por un programa externo denominado códec. Es decir, el audio y el video contenidos en el AVI pueden estar en cualquier formato. Por eso se le considera un formato contenedor. [9]

1.5 Formato AVI.

1.5.1 Reseña histórica.

Los archivos AVI son un caso especial de archivos RIFF (Resource Interchange File Format o Formato de Archivos para el Intercambio de Recursos) un formato de propósito general para el intercambio de datos multimedia. [12]

El formato AVI fue definido por Microsoft para su tecnología Video for Windows en 1992. Posteriormente fue mejorado mediante las extensiones de formato del grupo OpenDML de la compañía Matrox. Estas extensiones están soportadas por Microsoft, aunque no de manera oficial. [9]

Han existido dos versiones de formatos AVI: El primero que tenía algunas limitantes y la segunda versión que eliminó dichas limitantes, aunque ocasionó archivos gigantescos de video. Sólo existen dos tipos generales de AVI: los basados en Video for Windows (los primeros en aparecer) y los basados en DirectShow (originalmente ActiveMovie). [8]

1.5.2 ¿Cómo funciona?

Para que todos los flujos de datos de video y de audio puedan ser reproducidos simultáneamente es necesario que se almacenen de manera entrelazada. De esta manera, cada fragmento de archivo tiene suficiente información como para reproducir unos pocos fotogramas junto con el sonido correspondiente.

Obsérvese que el formato AVI admite varios flujos de datos de audio, lo que en la práctica significa que puede contener varias bandas sonoras en varios idiomas. Es el reproductor multimedia quien decide cuál de estos flujos debe ser reproducido, según las preferencias del usuario.

Los archivos AVI se dividen en fragmentos bien diferenciados denominados chunks. Cada chunk tiene asociado un identificador denominado etiqueta FourCC. El primer fragmento se denomina cabecera y su papel es describir meta-información respecto al archivo, por ejemplo, las dimensiones de la imagen y la velocidad en fotogramas por segundo. El segundo chunk contiene los flujos entrelazados de audio y video. Opcionalmente, puede existir un tercer chunk que actúa a modo de índice para el resto de chunks. [9]

1.5.3 Problemas más comunes.

Uno de los problemas más comunes es la falta de un estándar claro. Aunque parezca mentira, AVI no es un formato claro de almacenamiento de video, es más un formato de propósito general. La especificación del AVI lo único que dice es que si vamos a almacenar un video con ese formato, simplemente le pongamos una cabecera para identificar el tipo de fichero que es y que tenemos que almacenar un trozo de video y uno de audio a continuación y así sucesivamente para tener más o menos esto en el fichero:

CABECERA-VIDEO-AUDIO-VIDEO-AUDIO-VIDEO-AUDIO-...

El problema viene en que el trozo de video y de audio puede ser de cualquier tipo. Es decir, que no se especifica qué formato ha de tener el video y el audio.

Un AVI puedes crearlo con un formato de video y otro de sonido. El problema es que si no tenemos el códec adecuado en nuestro sistema, simplemente no funcionará. Si por ejemplo la película está en MPG4-OGG y no tenemos el códec OGG, pues veremos la película pero no la oiremos. Si la película está en Xvid-MP3, pues oiremos la película pero no la veremos. Si la película está en Xvid-OGG ni la veremos ni la oiremos (siempre que no tengamos los códec Xvid y OGG). [11]

1.5.4 Limites de tamaño de los archivos.

Cuando trabajamos con videos digitales a menudo hay que manejar archivos AVI de varios gigas de tamaño. Sin embargo hasta hace relativamente poco los sistemas no estaban diseñados para manejar tales tamaños y por ello muchas personas han tenido y tienen problemas para la creación de archivos AVI de larga duración.

Los límites más comunes en la actualidad son los 2 y 4 gigas. Estas barreras están causadas por dos factores: los límites del formato AVI estándar y el sistema de archivos del sistema operativo.

Como se ha dicho, muchas aplicaciones, especialmente las más antiguas, usan arquitectura basada en Video for Windows, lo que limita el tamaño de los archivos AVI a 2 GB. Los AVI estándar pueden llegar hasta los 4 gigas usando otros sistemas como DirectShow. [12]

1.5.5 Cómo se reproduce un archivo AVI.

Para reproducir un archivo AVI es necesario lo siguiente:

- Un intérprete del formato AVI.
- El códec de video para interpretar el flujo de video.
- El códec de audio para interpretar el flujo de audio.

La etiqueta FourCC permite identificar el códec necesario para interpretar un flujo de audio o video. Cada códec tiene asociados el conjunto de etiquetas que es capaz de reproducir. De esta manera, el intérprete de video es capaz de elegir el códec pertinente sin intervención del usuario.

El reproductor consecutivamente lee fragmentos del archivo AVI. Después separa cada uno de los flujos de audio y video que se encuentran entrelazados en el archivo. Cada uno de estos flujos, una vez separados, se almacenan en un buffer de memoria y se pasan al códec correspondiente. El códec de video devuelve otro buffer que contiene cada uno de los fotogramas a reproducir. El códec de audio retorna otro buffer con la muestra digital de sonido a reproducir. Con esta información, el reproductor solamente tiene que sincronizar los fotogramas y el sonido y reproducirlos a la velocidad adecuada. [9]

1.6 Técnica para trabajar con archivos de video AVI.

Para trabajar con archivos de video de extensión (.avi), generalmente se abre el archivo directamente, no se emplea ninguna técnica para codificar el archivo de video o cambiar su extensión con el objetivo de proteger los derechos del autor, aunque es posible realizar estas operaciones.

Por lo general un fichero (.avi) guarda solamente información de audio y de video, se utilizan arreglos para almacenar esta información. Lo importante es especificar qué tipo de información se cargará, si pertenece a video o audio. Una vez que se conoce el tamaño del arreglo es posible hacer ciclos.

Para facilitar abrir y reproducir los ficheros AVI, se pueden utilizar funciones ya existentes, por lo que la información es obtenida simplemente por el empleo de una función y no es necesario manejar la descompresión del video.

La información de audio generalmente no está comprimida, por lo que lo primero será determinar el tamaño de los datos de audio y cargarlos en un buffer. Luego se manda a sonar.

La información del video que se maneja, es principalmente datos como las dimensiones del AVI (ancho y altura), el número del último frame, así como cuántos milisegundos se muestran cada frame para mantener la velocidad.

Cuando el sistema sea inicializado, podemos hacer un ciclo que arranca y muestra cada frame de video, estos se guardan en un buffer y se convertirán cada uno en una imagen.

Solamente hay que especificar el formato en el que se almacena inicialmente y el formato en el que se convertirán los frames de la animación dependiendo del sistema que se empleará para pintar.

Se organizan los frames de manera que en la memoria solo se necesita uno a la vez, permitiendo la reproducción de archivos grandes.

Windows guarda los colores RGB como BRG por lo que es necesario corregir el problema, por el que los colores se muestran intercambiados, rojo con azul en el modelo RGB.

Si se está haciendo uso de la biblioteca gráfica OpenGl, se requiere que el tamaño de las texturas sea potencia de 2, por lo que se necesita una forma rápida de cambiar el video a un formato que podamos usar como textura. O utilizar para texturizar una función de las extensiones de OpenGL que permita que se utilice texturas que no sean potencia de 2.

Se determina por cual frame comenzar y haciendo uso de un temporizador en milisegundos se puede calcular la diferencia entre los frames para obtener una velocidad constante de reproducción.

Luego se puede dibujar directamente.

Cuando todos los marcos han sido procesados, es importante cerrar el sistema de descompresión. [20]

1.7 Tipos comprimidos / descomprimidos.

Una vez que la señal de video compuesta ha sido digitalizada y convertida en RGB, lo más normal es guardar esa información en algún soporte de los que habitualmente utilizan ordenador. Para poder hacerlo, se debe "capturar" esa imagen.

Se puede seguir dos procesos:

- Grabar una simple imagen en particular en cualquiera de los formatos de las imágenes existentes.
- Grabar un trozo entero de película, es decir, realizar una captura en tiempo real. Para poder realizar esta operación es necesario guardar el conjunto de imágenes y sonido en algún formato que será posteriormente recuperable por el ordenador. [13]

Como hemos dicho para cada punto de la imagen se le asigna un determinado número de bits que representarán el color de dicho punto. Si la imagen es en blanco y negro, bastará un bit para representarlo, mientras que para 256 colores serán necesarios 8 bits. De esta forma tendremos la imagen digitalizada, pero almacenar esta información dependerá del número de pixeles que utilicemos por imagen. Por ejemplo una imagen de 640 x 480 puntos con 256 colores ocupa 300 Kb, y si tenemos una secuencia de video a 25 fotogramas por segundo significaría que un solo segundo ocuparía 7.500 Kb. Y todo esto sin contar el audio.

La información de video compuesta de esta manera posee una cantidad tremenda de información, por lo que, para transmisión o almacenamiento, se requiere de la compresión de la imagen.

La compresión del video generalmente implica una pérdida de información y una consecuente disminución de calidad. Pero esto es aceptable porque los algoritmos de codificación están diseñados para descartar la información redundante o que no es perceptible por el ojo humano. Aunque sabemos que la calidad del video es inversamente proporcional al factor de compresión.

La compresión es un arma de doble filo, ya que el video comprimido es más sensible a los errores. Un error en video comprimido puede hacer ilegible la imagen, con lo que se añade redundancia para recuperar esa información. [4]

1.7.1 Códecs de Video.

Un códec de video es un programa que permite comprimir y descomprimir video digital. Normalmente los algoritmos de compresión empleados conllevan una pérdida de información. [14]

La técnica de compresión de video consiste de tres pasos fundamentalmente, primero el preprocesamiento de la fuente de video de entrada, en el cual se realiza el filtrado de la señal de entrada para remover componentes no útiles y el ruido que pudiera haber en esta. El segundo paso es la conversión de la señal a un formato intermedio común, y por último el paso de la compresión. [4]
La digitalización y la compresión pueden darse conjuntamente y en tiempo real para facilitar la comunicación y la interacción.

El problema que se pretende acometer con los códec es que la información de video es bastante ingente en relación a lo que un ordenador normal es capaz de manejar. Es así como un par de segundos de video en una resolución apenas aceptable puede ocupar un lugar respetable en un medio de almacenamiento típico (disco duro, cd, dvd) y su manejo (copia, edición, visualización) puede llevar fácilmente a sobrepasar las posibilidades de dicho ordenador o llevarlo a su límite.

Su finalidad es obtener un almacenamiento substancialmente menor de la información de video. Esta se comprime en el momento de guardar la información hacia un archivo y se descomprime, en tiempo real, en el momento de la visualización. Se pretende, por otro lado, que éste sea un proceso transparente para el usuario, es decir, que éste no intervenga o lo haga lo menos posible.

Existe un complicado equilibrio entre la calidad de video, la cantidad de datos necesario para representarlo (también conocida como tasa de bits), la complejidad de los algoritmos de codificación y decodificación, la robustez frente a las pérdidas de datos y errores, la facilidad de edición, la posibilidad de acceder directamente a los frames, y otros factores. [14]

Durante el proceso de decodificar antes de poder ser visualizadas por el usuario se puede producir lo que se llama "video fantasma" o suavización de imagen, que es la forma con la que los códecs compensan los elevados flujos de información. Cuando ocurre esto, el códec comprime la información reduciendo el "framerate" (número de imágenes por segundo), el cual puede hacer que los movimientos rápidos parezcan borrosos. El códec también modifica la resolución para comprimir la

información lo cual puede hacer que la imagen se vea desplazada. Entonces, para reducir estos efectos, se disminuye el flujo de información visual.

Los códecs se optimizan para conseguir la mayor calidad posible en bajos índices de transferencia. Son usados para codificar el video en tiempo real o pregrabado. [4]

Por intereses comerciales no usan un mismo códec para todo. En primer lugar están los derechos de patente del formato de sonido-video. Y en segundo lugar están los intereses de las grandes compañías audiovisuales que no desean fomentar un formato con el que la gente pueda ver sus películas). [11]

1.8 Tendencia a utilizar Videos.

La tendencia más marcada en la actualidad, para usar videos, es como presentaciones en las aplicaciones gráficas, sobre todo en video-juegos y simuladores. Como es el caso de los juegos Need for Speed (ver anexo 1), MBP (ver anexo 2) y simuladores como VirtualGT (ver anexo 3) y Ace Combat 6 (ver anexo 4). En otros se utilizan como transición entre misiones por ejemplo en juegos como Proyecto IGI (ver anexo 5) y en ocasiones se pueden generar de acuerdo a situaciones específicas como es el caso de Fifa2005 (ver anexo 6).

En el caso de los juegos los videos constituyen una forma agradable y entretenida de conocer la historia del juego, es una forma más de cautivar a los jugadores introduciéndolos cada vez más en un mundo real o imaginario. Mientras que en los simuladores los videos son empleados sobre todo en las presentaciones y en ocasiones se utilizan para dar algún tipo de instrucción. En ambos casos los videos constituyen una manera de introducir a los usuarios en distintos tipos de misiones o situaciones específicas.

1.9 Scene Toolkit.

Desde los primeros pasos de la Facultad 5 en el desarrollo de aplicaciones de Realidad Virtual, se planteó la necesidad de crear un grupo de herramientas de apoyo a los programadores de aplicaciones finales, acorde con la tendencia mundial de desarrollo de los llamados “engines” o motores gráficos.

SceneToolKit es la primera y una de las herramientas brindadas por el Proyecto de Herramientas de Desarrollo para Sistemas de Realidad Virtual, y surge como una respuesta a esta necesidad, teniendo

como objetivo básico agrupar las funcionalidades comunes a cualquier sistema de Realidad Virtual, de manera que se les facilite el trabajo a los programadores de juegos y simuladores a través de la reutilización de código.

Esta herramienta, en desarrollo actualmente, permite no solamente la visualización de los entornos sintéticos sino además la aplicación de leyes físicas y matemáticas, animaciones, etc., usando las librerías gráficas OpenGL y DirectX, sobre plataforma Windows y Linux. A la vez, se retroalimenta con las necesidades de los programadores que la utilizan, así como de sus investigaciones. [15]

1.9.1 Arquitectura de la SceneToolKit.

La implementación de la herramienta se hizo en C++ utilizando como IDE el VisualStudio 2003 sobre Windows, y el CodeBlocks (versión svn 3586) sobre Linux.

Incluye las estructuras de la std para el trabajo con estructuras como listas y otras. Esta está dividida en 3 capas fundamentales: una capa Engine donde se encuentra todo el procesamiento importante de carga y manejo de objetos; una capa Renderer donde se define con qué librería gráfica dibujar los entornos (OpenGL (GLUT y WGL) y DirectX por el momento), y que permite añadir nuevas librerías por parte de los usuarios de la herramienta en caso de que lo necesiten; y una capa Application donde se hace el manejo de eventos del SO específico (Windows y Linux para esta versión), donde el usuario de la herramienta puede desarrollar su propia interfaz para cualquier SO. [15]

1.10 Biblioteca AVI file para Linux.

Este proyecto intenta proporcionar utilidades relacionadas con las multimedia para Linux. En el momento de su creación en mayo del 2000 su objetivo primario era la captura de video y la recomprensión, aplicaciones que trabajarían con el formato más popular de archivo AVI y los métodos de compresión de datos más recientes (Indeo Video y las variaciones de MPEG-4 para la compresión de imágenes y MPEG Layer-3/Windows Media Audio para el sonido). La idea principal del proyecto estaba en la utilización de bibliotecas dinámicas Win32 en el ambiente Linux.

Sin embargo, se ha hecho más popular debido a un producto que provee - un reproductor de películas AVI que puede reproducir películas codificadas con DivX;-) en Linux con buen funcionamiento y estabilidad. Desde entonces la mayor parte del trabajo fue hecho en esta dirección. Ahora soporta una amplia gama de códecs (compresores/descompresores) como DivX;-), Indeo Video, I263, y otros, es capaz de mostrar subtítulos y configuración de la salida del video usando YUV cuando es necesario el

soporte del hardware y el sistema operativo está disponible. Es también capaz de reproducir archivos en el formato de ASF.

El proyecto no intenta proporcionar un framework universal para Linux, ni hacer mucho más que lo que actualmente hace. Además, esto es sobre todo un trabajo de prueba-de-concepto. Este fue el primer proyecto que introdujo la idea de usar Windows DLLs y muy limitado el subconjunto de APIs Win32 para audio/video (de)compresión en ambientes Unix, la idea que ya es reutilizada en varios otros proyectos de software. Este es el primer proyecto que amplía la idea del uso de decodificadores DirectShow de audio/video para el mismo propósito (para la emulación de ambientes DirectShow / DCOM desde el punto de vista de un decodificador). Este es el único proyecto que incluye un reproductor para archivos en Formato Avanzado. Todos estos rasgos son esenciales para el completo y moderno mundo de la multimedia, y con esperanza se hará disponible en un futuro cercano en arquitecturas como Arts (KDE) o GStreamer (Gnome).

Aparte de la reutilización de las ideas, el proyecto es usado en algunos reproductores para Linux, como XMMS, XTheater o LAMP.

El código de este proyecto es distribuido bajo la versión 2 de General Public License.

Básicamente esto quiere decir que se puede hacer cualquier producto con este código e incluso modificar el código, pero si se quiere redistribuir o cualquiera de sus derivados, es necesario hacerlo bajo GPL.

Las cuestiones legales que cubren Win32 DLLs que acompaña el código original son un poco más complicadas. Estas DLLs están libremente disponibles en Internet. Para aquellas DLLs que vienen con la licencia, sus titulares de derecho de autor permiten la utilización gratis de ellas mientras no se desensamblen. En algunos casos, como Indeo Video, permiten explícitamente incluir estos archivos en otros proyectos conforme a restricciones mencionadas. Muchas DLLs están disponibles sin necesidad de aceptar ningún acuerdo de licencia, ejemplo (DivX;-), (DirectShow códecs), que obviamente quiere decir que cualquier tipo de actividad con ellos es aceptable.

La biblioteca está escrita sobre todo por un antiguo estudiante de universidad con conocimientos en el área de la Física Aplicada en sus ratos libres. [16]

1.11 Funciones de la API de Windows.

La Interfaz de Programación de Aplicaciones, cuyo acrónimo en inglés es API (Application Programming Interface), es un conjunto de funciones residentes en bibliotecas (generalmente dinámicas) que permiten que una aplicación corra bajo el sistema operativo Windows.

Debido a su estrecha relación con el desarrollo de software, los programas en sus especificaciones generalmente traen implícita la versión de la API del sistema operativo, mediante diversas nomenclaturas tales como la versión específica del sistema operativo (para Windows 98, por ejemplo), o la versión del conjunto de librerías (Plataforma Win32, etc.).
La primera versión de esta API fue de 16 bits, y llamada Win16. Sólo se utilizaba en las versiones de 16 bits de Windows.

En su nueva versión 32 bits, se incrementó el número de APIs disponibles para los sistemas operativos Microsoft Windows. Microsoft proporciona un SDK (kit de desarrollo de aplicaciones) en el que se incluyen la documentación y las herramientas necesarias para que los programadores puedan crear sus aplicaciones y aprovechar los recursos del sistema.

Las versiones modernas de Windows utilizan la API de 32 bits llamada Win32. Está compuesta por funciones en C almacenadas en librerías de enlace dinámico (DLL), especialmente en las del núcleo.

Aunque la implementación de Microsoft tiene derechos de autor, generalmente se acepta que otras empresas puedan emular Windows proporcionando APIs idénticas, sin que implique violación de derechos de autor.

La extensión 64 bits de la versión 32 bits se llama Win64.

Para desarrollar programas que funcionen en Windows se necesita un compilador que maneje las DLLs y objetos COM específicos de Microsoft, así como también un cierto número de archivos de cabecera de C (header files, .h) que definen las interfaces de las DLLs. [1]

La ventaja de utilizar las API de Windows en el código es que pueden ahorrar tiempo porque contienen numerosas funciones útiles ya escritas y listas para utilizar. La desventaja es que puede resultar difícil trabajar con las API de Windows y pueden ser implacables cuando las cosas van mal.

Las API de Windows representan una categoría especial de interoperabilidad. No utilizan código administrado, no tienen bibliotecas de tipos integradas y utilizan tipos de datos que son diferentes a los que se utilizan en Visual Studio. Debido a estas diferencias y a que las API de Windows no son objetos COM, la interoperabilidad con las API de Windows y .NET Framework se lleva a cabo mediante la invocación de la plataforma o PInvoke. [2]

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se proponen las soluciones técnicas para la creación de funcionalidades para la manipulación de videos digitales en Entornos de Realidad Virtual y soluciones específicas para lograr su integración a la biblioteca gráfica utilizada. Además se comienza a tener una visión del sistema a realizar y se dan los primeros pasos en su concepción práctica, desde el punto de vista de qué es lo que debe hacer el sistema.

2.1 Soluciones Técnicas.

Dada la necesidad de integración del módulo de video a la herramienta en desarrollo (SceneToolKit) y a otras aplicaciones gráficas o de multimedia, así como la creciente tendencia hacia el desarrollo de productos multiplataforma y con características de software libre, se propone el uso de OpenGl, como biblioteca para la visualización de los videos en un mundo 3D.

Se trabajará básicamente orientado a lograr una biblioteca multiplataforma que brinde las funcionalidades requeridas para la utilización de videos.

Después del análisis de los diferentes formatos de video se ha planteado el uso de los ficheros de video de extensión .avi, ya que se trata de un formato de archivo que actúa como contenedor de flujos de datos de audio y video, es decir, el audio y el video contenidos en el AVI pueden estar en cualquier formato, codificados con cualquier códec. Además es el formato más extendido para el manejo de datos de audio/video en una PC. Este formato es compatible con gran número de plataformas como Windows, Linux y Macintosh, esto será un paso más en la portabilidad del módulo hacia otras plataformas.

En tiempo de ejecución pueden aparecer errores tan sencillos como que un fichero de video está dañado o que no está en la ubicación esperada. Para esto el módulo implementará un sistema de tratamiento de errores, facilitando su manejo.

Para el Modelado Visual se utilizará la herramienta de desarrollo de software Rational Rose la cual está basada en UML, la notación estándar para arquitectura de software.

Se codificará en C++ utilizando como IDE Code Block 8.02, y se hará uso de funciones de la API de Windows para leer la información del fichero de video cuando se esté codificando sobre la plataforma Windows y en el caso de Linux se empleara la biblioteca avifile-0.7 con licencia GPL.

2.2 Modelo del Dominio.

Se propone el modelo de dominio correspondiente a la Figura 1. En el que se representan los conceptos fundamentales asociados al trabajo con videos digitales. Se representa el reproductor (que reproduce el video) y este tiene un formato determinado y ocupa una posición en un mundo virtual. Además, el video está compuesto por dos tipos de información: audio e imagen, los cuales pueden estar codificados indistintamente. En el caso del audio puede presentar uno (mono) o dos canales (estéreo).

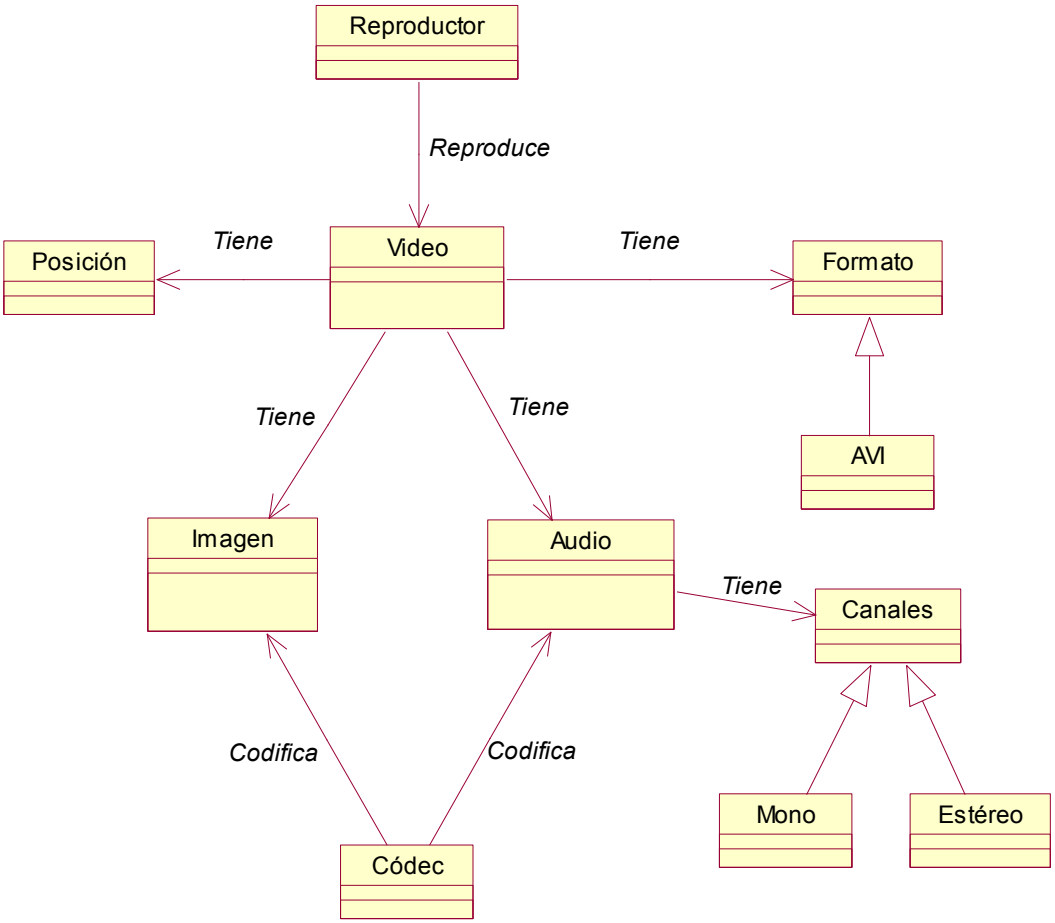

Figura 1: Modelo del Dominio.

2.2.1 Glosario de términos del dominio.

Reproductor: Programa o aplicación capaz de reproducir un video.

Video: Fichero con una extensión determinada que contiene datos de audio e imagen.

Posición: Ubicación vectorial en un ambiente virtual (x, y, z).

Formato: Conjunto de reglas o especificaciones mediante las cuales se pueden organizar datos.

AVI: Es un tipo de archivo contenedor, su nombre proviene de que la información del video y audio están "entrelazados" dentro del contenedor.

Audio: Tipo de información contenido en un fichero de video, se refiere al sonido.

Imagen: Tipo de información contenido en un fichero de video, se refiere al video.

Códec: Es un software que consta de una serie de algoritmos que permiten codificar y decodificar contenidos multimedia.

Mono: Sistema de registro y reproducción que utiliza solo un canal de información sonora.

Estéreo: Sistema de registro y reproducción que utiliza dos canales de información sonora: izquierdo y derecho.

2.3 Reglas del negocio

Las reglas del negocio -que no son más que parámetros o restricciones por los que se rige el sistema-, se dictaminaron debido al intercambio con los desarrolladores de la herramienta SceneToolKit , siendo necesario considerar las mismas para el desarrollo de la solución informática.

1. Los videos puede tener cualquier dimensión (ancho por altura).
2. Los videos pueden estar codificados con cualquier códec.
3. Leer ficheros de video no implica leer audio y video, puede ser solo uno de ellos.

2.4 Requerimientos del Sistema.

A continuación se hace referencia a los requisitos funcionales y no funcionales más significativos capturados para el desarrollo de la biblioteca.

2.4.1 Requisitos funcionales.

1. Inicializar sistemas para leer ficheros de video.
2. Abrir ficheros de video en modo de lectura.
3. Mostrar mensaje de error si no se puede abrir un fichero de video.
4. Leer datos de las imágenes del fichero de video.
5. Mostrar mensaje de error si los datos de la imagen no pueden ser leídos.
6. Leer datos del audio del fichero de video.
7. Mostrar mensaje de error si los datos de audio no pueden ser leídos.
8. Almacenar datos de las imágenes.
9. Almacenar datos del audio.

10. Inicializar la biblioteca gráfica (OpenGl).
11. Separar la carga del video de su ejecución.
12. Leer información del frame actual del video.
13. Almacenar en memoria la imagen referente al frame actual del video.
14. Pasar al siguiente frame del video.
15. Actualizar la imagen en memoria dependiendo del frame del video.
16. General textura dependiendo de la imagen en memoria.
17. Leer coordenadas de textura.
18. Crear vértices.
19. Habilitar dispositivo de audio.
20. Llenar un buffer de sonido con los datos de audio.
21. Reproducir buffer de sonido.
22. Comprobar el tiempo transcurrido desde la última llamada.
23. Sincronizar la actualización de los frames del video, dependiendo de los frames por segundos que presente el video.
24. Comenzar la reproducción del video (Play).
25. Pausar la reproducción del video (Pause).
26. Detener la reproducción del video (Stop).
27. Conocer el estado en que se encuentra el video (reproduciéndose, pausado o detenido).
28. Cerrar el dispositivo de audio.
29. Cerrar el fichero de video.

2.4.2 Requisitos no funcionales.

Usabilidad.

• Fácil para el programador.

Portabilidad.

• Multiplataforma.

Soporte.

• Va a contar con Ayuda y especificaciones de uso.

Software.

• Sistema Operativo Windows o Linux.

• Es necesario instalar los códecs con que este codificado el video.

Hardware.

• Es necesaria una PC con CPU Intel Pentium IV, 1 Gb de HDD, 256 Mb de memoria RAM.

2.5 Diagrama de Casos de Uso.

Se propone el diagrama de casos de uso que representa la Figura 2, el cual agrupa todos los requisitos funcionales.

El caso de uso “Gestionar Video” se refiere a las funciones de lectura (abrir y cerrar), del fichero de video. Con el caso de uso “Texturizar Polígono” se representa el hecho de crear un polígono texturizado con la imagen de un video, mientras que en el caso de uso “Reproducir Audio” se logra agrupar las funciones para reproducir el sonido referente a un video. Por último, el caso de uso “Controlar Video” permite controlar en todo momento el estado en el que se encuentra el video.

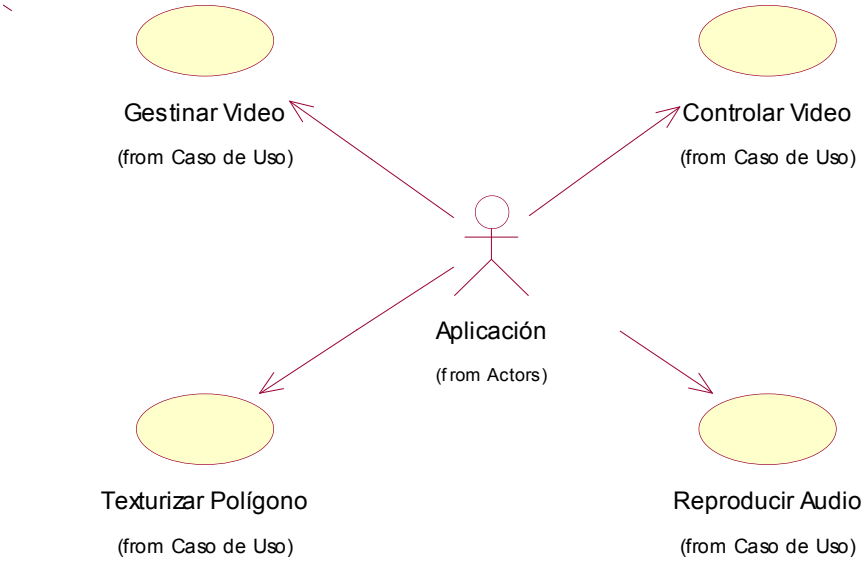

Figura 2: Diagrama de Casos de Uso del Sistema.

2.6 Expansión de los Casos de Uso.

2.6.1 Definición de los actores.

Actores	**Justificación**
Aplicación	La aplicación que utilice la biblioteca es la que se beneficiará con las funcionalidades que brinda este módulo para video.

Tabla 1: Definición de los actores.

2.6.2 Casos de Uso expandidos.

Nombre del caso de uso	Gestionar Video
Actores	Aplicación
Propósito	Cargar el fichero de video, leer la información referente al video y cerrarlo una vez finalizado el trabajo con esta información.
Resumen: El caso de uso se inicia cuando la Aplicación solicita cargar un video, se inicializan los sistemas que se emplearán para leer la información del video, entonces se procede a la carga de los datos del fichero de video. El caso de uso termina cuando se solicita cerrar el fichero de video.	
Precondiciones	Que exista un fichero de video.
Referencias	R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R29
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1- Solicita cargar un video especificando la dirección.	2- Se inicializan los sistemas que se emplearán para leer la información del video.
	3- Se inicializa la biblioteca gráfica.
	4- Se abre el archivo .avi directamente en modo de lectura.
	5- Se cargan los datos de audio del fichero de video.
	6- Se cargan los datos de la imagen del fichero de video.
	7- Se cierra el fichero de video.
Curso Alternativo de los eventos	
	4.1- Si el fichero no existe muestra un mensaje de error.
	5.1- Si la información no es válida muestra un mensaje de error.
	6.1- Si la información no es válida muestra un mensaje de error.
Poscondiciones	La información sobre el audio y la imagen correspondiente al fichero de video son almacenadas en memoria.

Tabla 2: Expansión del Caso de Uso (Gestionar Video).

Nombre del caso de uso	Controlar Video
Actores	Aplicación
Propósito	Mantener un control sobre el estado en que se encuentra el video.

Resumen: Se inicia el caso de uso cuando la Aplicación solicita cambiar de estado el video. Entonces se procede a variar a los estados deseados, el caso de uso termina cuando el video alcanza el estado deseado. (playing , paused, stopped).	
Precondiciones	Que la información del fichero de video se haya leído correctamente.
Referencias	R24,R25,R26,R27
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
Sección: Play.	
1-Se solicita comenzar la reproducción del video.	2- Se varía el estado del video.
	3- Se comienza a reproducir el sonido.
	4- Se comienza la actualización de la imagen en memoria, con la imagen correspondiente al próximo frame del video.
Sección: Pause.	
1-Se solicita pausar la reproducción del video.	2- Se varía el estado del video.
	3- Se pausa la reproducción del sonido.
	4- Se pausa la actualización de la imagen en memoria.
Sección: Stop.	
1-Se solicita detener la reproducción del video.	2- Se varía el estado del video.
	3- Se detiene la reproducción del sonido.
	4- Se detiene la actualización de la imagen en memoria.
Sección: Estado.	
1-Se solicita conocer el estado en que se encuentra el video.	2- Se devuelve el estado en que se encuentra el video. (playing, paused, stopped).
Curso Alternativo de los eventos	
	1.1- Si la información del fichero de video no se ha leído correctamente, la operación no se lleva a cabo y finaliza el caso de uso.
Poscondiciones	El video toma el estado asignado.

Tabla 3: Expansión del Caso de Uso (Controlar Video).

Nombre del caso de uso	Texturizar Polígono
Actores	Aplicación
Propósito	Texturizar un polígono de un mundo 3d con la imagen de un video.
Resumen: Se inicia el caso de uso cuando la Aplicación solicita crear un polígono texturizado con la imagen de un video específico. Se actualiza cada cierto tiempo, la imagen del frame del video en memoria, dependiendo de la cantidad de frames por segundos del video. Se genera una nueva textura correspondiente a la imagen en memoria. Se crea el polígono, texturizado con las coordenadas de la textura generada. El caso de uso termina cuando se obtiene un polígono texturizado.	
Precondiciones	Que la información de la imagen del video haya sido leída correctamente.
Referencias	R12,R13,R14,R15,R16,R17,R18,R22,R23
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1- Se especifica el video que se desea visualizar.	
1.1- Se especifica las coordenadas del polígono que se texturizará con las imágenes del video.	2- Se comprueba el tiempo transcurrido desde la última llamada.
	3- Se crea en un buffer la imagen de un frame del video, el cual se actualiza cada cierto tiempo dependiendo de la cantidad de frames por segundos del video.
	4- Se genera una textura a partir de la imagen en memoria.
	5- Se crean los vértices del polígono. Mientras se leen las coordenadas de la textura generada.
Curso Alternativo de los eventos	
	1.2- Si la información del fichero de video no se ha leído correctamente, la operación no se lleva a cabo y finaliza el caso de uso.
Poscondiciones	Se obtiene un polígono texturizado con las imágenes de un video.

Tabla 4: Expansión del Caso de Uso (Texturizar Polígono).

Nombre del caso de uso	Reproducir Audio
Actores	Aplicación
Propósito	Reproducir el audio de un video.
Resumen: Se inicia el caso de uso cuando la Aplicación solicita que se reproduzca el audio de un video específico. Se habilita el dispositivo de audio, se crea un buffer con la información del audio del video y se reproduce. El caso de uso termina cuando finaliza la reproducción del audio o cuando la Aplicación lo solicite.	
Precondiciones	Que la información del audio del video haya sido leída correctamente.
Referencias	R19,R20,R21,R28
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1- Se solicita reproducir el audio de un video.	2- Se Habilita el dispositivo de audio.
	3- Se Llena un buffer de sonido con los datos de audio.
	4- Se comienza a reproducir el buffer de sonido.
	5- Se cierra el dispositivo de audio.
Curso Alternativo de los eventos	
	1.2- Si la información del fichero de video no se ha leído correctamente, la operación no se lleva a cabo y finaliza el caso de uso.
Poscondiciones	Se reproduce el audio del video.

Tabla 5: Expansión del Caso de Uso (Reproducir Audio).

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN

Introducción

En el presente capítulo se muestra el diseño de clases del sistema propuesto, en donde se definen las responsabilidades de estas y sus relaciones. Además se muestran los diagramas de secuencia por casos de uso que intervendrán en el primer ciclo de desarrollo de software, facilitando el entendimiento de la biblioteca.

Esta etapa del proyecto se conforma con el paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .h y .cpp correspondientes a la implementación en C++. También se da a conocer la nomenclatura de las versiones y los estándares de codificación.

3.1 Diagrama de Diseño de clases.

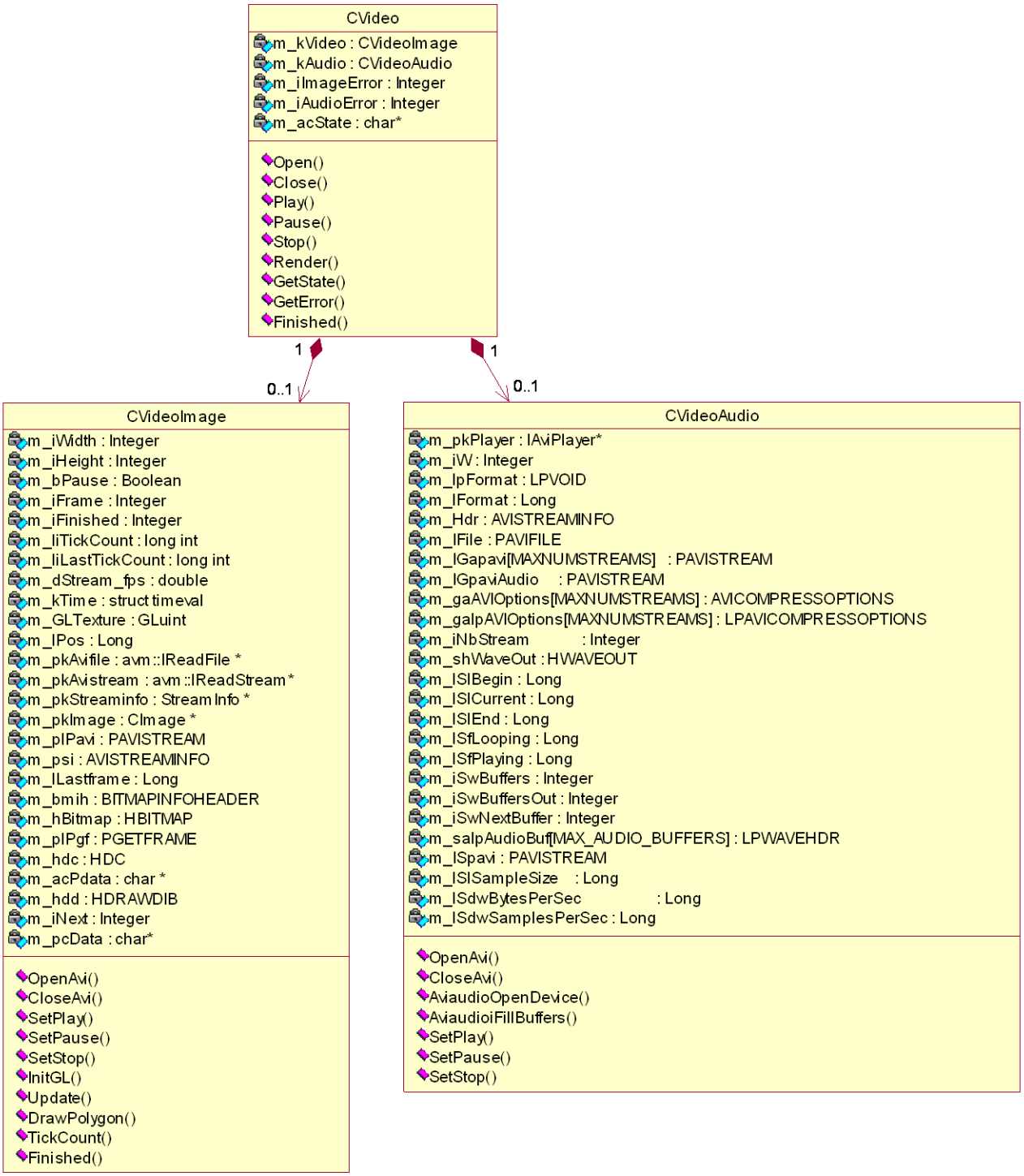

Figura 3: Diagrama de Diseño de clases.

El diagrama de diseño de clases que se propone, correspondiente a la Figura 3, muestra la composición existente entre las clases que manipulan el audio “CVideoAudio” e imagen “CVideoImage” del video y la clase “CVideo”.

Un fichero de video puede soportar más de un audio pero para la confección de la biblioteca se consideró solo un audio. Además teniendo en cuenta la regla de negocio número tres, la cual plantea que leer ficheros de video no implica leer audio y video, puede ser sólo uno de ellos, por lo que se define como mínimo de ocurrencia “0” para el caso de ambas clases CVideoImage y CVideoAudio.

3.2 Descripción de las clases.

Nombre: CVideo	
Controladora	
Atributo	**Tipo**
m_kImage	CVideoImage
m_kAudio	CVideoAudio
m_iImageError;	int
m_iAudioError;	int
m_acState	char*
Para cada responsabilidad:	
Nombre:	Open (arg_acDir : char*)
Descripción:	Se manda a las clases encargadas de extraer el audio y la imagen del video que abran el fichero especificado. Luego se manda a detener la reproducción del video.
Nombre:	Close()
Descripción:	Se manda a las clases encargadas de gestionar la imagen y el audio del video a cerrar el video.
Nombre:	Play()
Descripción:	Se manda a las clases encargadas de controlar la imagen y el audio del video a comenzar a reproducir el video.
Nombre:	Pause()
Descripción:	Se manda a las clases encargadas de controlar la imagen y el audio del video a pausar la reproducción del video.

Nombre:	Stop()
Descripción:	Se manda a las clases encargadas de controlar la imagen y el audio del video a detener la reproducción del video.
Nombre:	Render (arg_GLXRightTop : GLfloat, arg_GLYRightTop : GLfloat, arg_GLZRightTop : GLfloat, arg_GLXLeftTop : GLfloat, arg_GLYLeftTop : GLfloat, arg_GLZLeftTop : GLfloat, arg_GLXLeftBottom : GLfloat, arg_GLYLeftBottom : GLfloat, arg_GLZLeftBottom : GLfloat, arg_GLXRightBottom : GLfloat, arg_GLYRightBottom : GLfloat, arg_GLZRightBottom : GLfloat)
Descripción:	Se manda a las clases encargadas de manejar la imagen del video a texturizar el polígono especificado.
Nombre:	GetError()
Descripción:	Retorna el tipo de error al cargar el fichero de video y extraerle la información de la imagen y el audio.
Nombre:	GetState()
Descripción:	Retorna el estado en el que se encuentra el video en todo momento (playing, paused, stopped).
Nombre:	Finished()
Descripción:	Retorna la cantidad de veces que el video ha llegado a su final.

Tabla 6: Descripción de la clase (CVideo).

Nombre: CVideoAudio	
Controladora	
Atributo	**Tipo**
m_pkPlayer	IAviPlayer*
m_iW	int
m_lpFormat	LPVOID
m_lFormat	long
m_Hdr	AVISTREAMINFO
m_lGapavi[MAXNUMSTREAMS]	PAVISTREAM
m_lGpaviAudio	PAVISTREAM
m_gaAVIOptions[MAXNUMSTREAMS]	AVICOMPRESSOPTIONS

m_galpAVIOptions[MAXNUMSTREAMS]	LPAVICOMPRESSOPTIONS
m_iNbStream	int
m_shWaveOut	HWAVEOUT
m_lSlBegin	long
m_lSlCurrent	long
m_lSlEnd	long
m_lSfLooping	long
m_lSfPlaying	long
m_iSwBuffers	int
m_iSwBuffersOut	int
m_iSwNextBuffer	int
m_salpAudioBuf[MAX_AUDIO_BUFFERS]	LPWAVEHDR
m_lSpavi	PAVISTREAM
m_lSlSampleSize	long
m_lSdwBytesPerSec	long
m_lSdwSamplesPerSec	long

Para cada responsabilidad:	
Nombre:	OpenAvi (arg_acDir : char*)
Descripción:	Se inicializa la biblioteca que se utiliza para leer el fichero de video. Se carga la información del audio del fichero de video especificado y se guarda en las estructuras creadas para su almacenamiento.
Nombre:	CloseAvi()
Descripción:	Se cierra la biblioteca que se utiliza para leer el fichero de video. Se limpian las estructuras que se utilizaron para almacenar la información del video.
Nombre:	AviaudioOpenDevice(arg_acPavi : char*)
Descripción:	Este método es solo para Windows. Se encarga de abrir el dispositivo de audio.
Nombre:	AviaudioiFillBuffers()
Descripción:	Este método es solo para Windows. Es llamado en el método OpenAvi (arg_acDir : char*), se encarga de llenar el buffer con la información del audio leída del fichero de video.
Nombre:	SetPlay()
Descripción:	Se manda a comenzar a reproducir el audio del video.

Nombre:	SetPause()
Descripción:	Se manda a pausar la reproducción del audio del video.
Nombre:	SetStop()
Descripción:	Se manda a detener la reproducción del audio del video.

Tabla 7: Descripción de la clase (CVideoAudio).

Nombre: CVideoImage	
Controladora	
Atributo	**Tipo**
m_kTime	struct timeval
m_bPause	bool
m_iFinished	int
m_GLTexture	GLuint
m_iFrame	int
m_iWidth	int
m_iHeight	int
m_lPos	long
m_dStream_fps	double
m_pkAvifile	avm::IReadFile *
m_pkAvistream	avm::IReadStream*
m_pkStreaminfo	StreamInfo *
m_pkImage	CImage *
m_liTickCount	long int
m_liLastTickCount	long int
m_plPavi	PAVISTREAM
m_Psi	AVISTREAMINFO
m_lLastframe	long
m_Bmih	BITMAPINFOHEADER
m_hBitmap	HBITMAP
m_plPgf	PGETFRAME
m_Hdc	HDC
m_acPdata	char *

m_Hdd	HDRAWDIB
m_iNext	int
m_pcData	char*
Para cada responsabilidad:	

Nombre:	OpenAvi (arg_acDir : char*)
Descripción:	Se manda a inicializar la biblioteca gráfica que se utiliza para visualizar la imagen del video. Se inicializa la biblioteca que se utiliza para leer el fichero de video. Se carga la información referente a la imagen del fichero de video especificado y se guarda en las estructuras creadas para su almacenamiento.
Nombre:	CloseAvi()
Descripción:	Se cierra la biblioteca que se utiliza para leer el fichero de video. Se limpian las estructuras que se utilizaron para almacenar la información del video.
Nombre:	SetPlay()
Descripción:	Se manda a comenzar la actualización de la imagen en memoria correspondiente a un frame del video.
Nombre:	SetPause()
Descripción:	Se manda a pausar la actualización de la imagen en memoria correspondiente a un frame del video.
Nombre:	SetStop()
Descripción:	Se manda a detener la actualización de la imagen en memoria correspondiente a un frame del video.
Nombre:	InitGL()
Descripción:	Se Inicializa la biblioteca gráfica. (OpenGl)
Nombre:	Update (arg_iTime : int)
Descripción:	Actualiza la imagen en memoria correspondiente a cada frame del video, dependiendo de los frames por segundos que tenga el video.
Nombre:	DrawPolygon(arg_GLXRightTop : GLfloat, arg_GLYRightTop : GLfloat, arg_GLZRightTop : GLfloat, arg_GLXLeftTop : GLfloat, arg_GLYLeftTop : GLfloat, arg_GLZLeftTop : GLfloat, arg_GLXLeftBottom : GLfloat, arg_GLYLeftBottom : GLfloat, arg_GLZLeftBottom : GLfloat, arg_GLXRightBottom : GLfloat, arg_GLYRightBottom : GLfloat, arg_GLZRightBottom : GLfloat)
Descripción:	Se comprueba el tiempo trascurrido desde la última llamada para actualizar la

	imagen en memoria del frame del video mediante la función TickCount(). Luego se genera una textura con la imagen en memoria. Se crean los vértices del polígono especificado mientras se leen las coordenadas de la textura generada.
Nombre:	TickCount()
Descripción:	Es el encargado de mantener un ritmo de llamadas constante a la función Update (arg_iTime : int), dependiendo de la cantidad de frames por segundo que tiene el video.
Nombre:	Finished()
Descripción:	Retorna la cantidad de veces que se ha llegado a procesar la imagen correspondiente al último frame del video.

Tabla 8: Descripción de la clase (CVideoImage).

3.3 Diagramas de Secuencia.

A continuación se ilustran los diagramas de secuencia vinculados a los Casos de Usos especificados.

El diagrama de secuencia “Controlar Video” correspondiente a la sección: Estado que se propone, en la Figura 4, muestra al actor Aplicación interactuando con la clase CVideo, cuando el actor desee conocer el estado en el que se encuentra el video (reproduciéndose, pausado o detenido) se devuelve la información referente al estado.

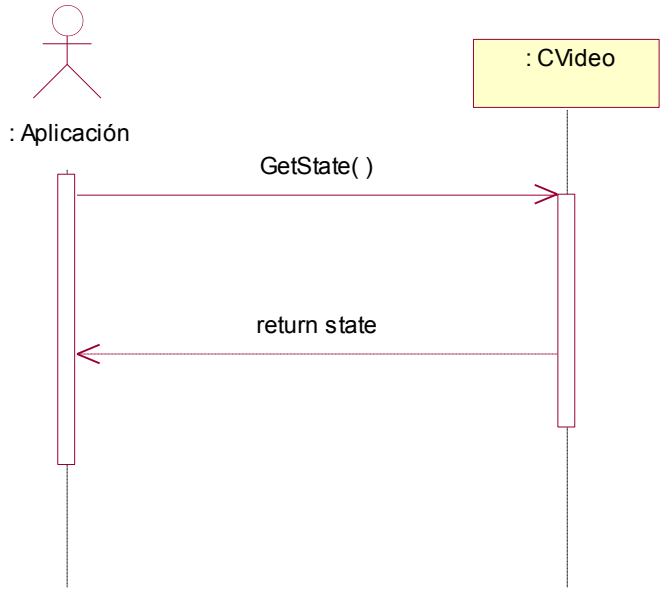

Figura 4: Diagrama de Secuencia Controlar Video Sección: Estado.

El diagrama de secuencia “Controlar Video” correspondiente a la sección: Pause que se propone, en la Figura 5, muestra al actor Aplicación interactuando con la clase CVideo, la cual envía, cuando el actor desee pausar el video, el mensaje de pausarse a las clases encargadas de manipular la información referente a la imagen y al audio respectivamente. Como resultado la reproducción del video es pausada.

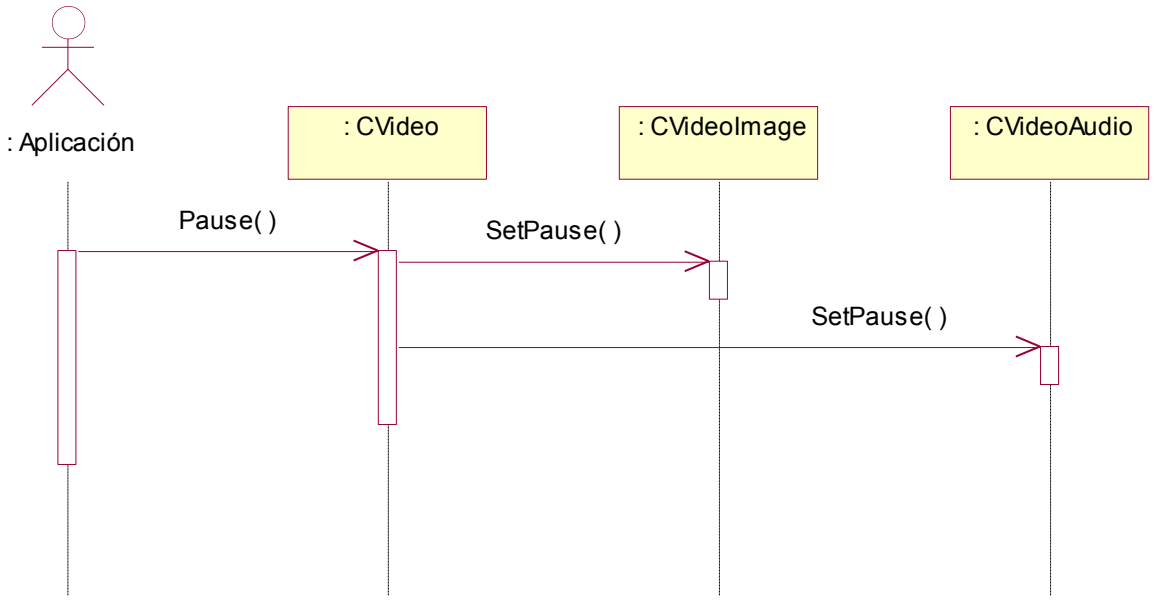

Figura 5: Diagrama de Secuencia Controlar Video Sección: Pause.

El diagrama de secuencia “Controlar Video” correspondiente a la sección: Play que se propone, en la Figura 6, muestra al actor Aplicación interactuando con la clase CVideo, la cual envía, cuando el actor desee reproducir el video, el mensaje de comenzar la reproducción a las clases encargadas de manipular la información referente a la imagen y al audio respectivamente. Como resultado se comienza la reproducción del video.

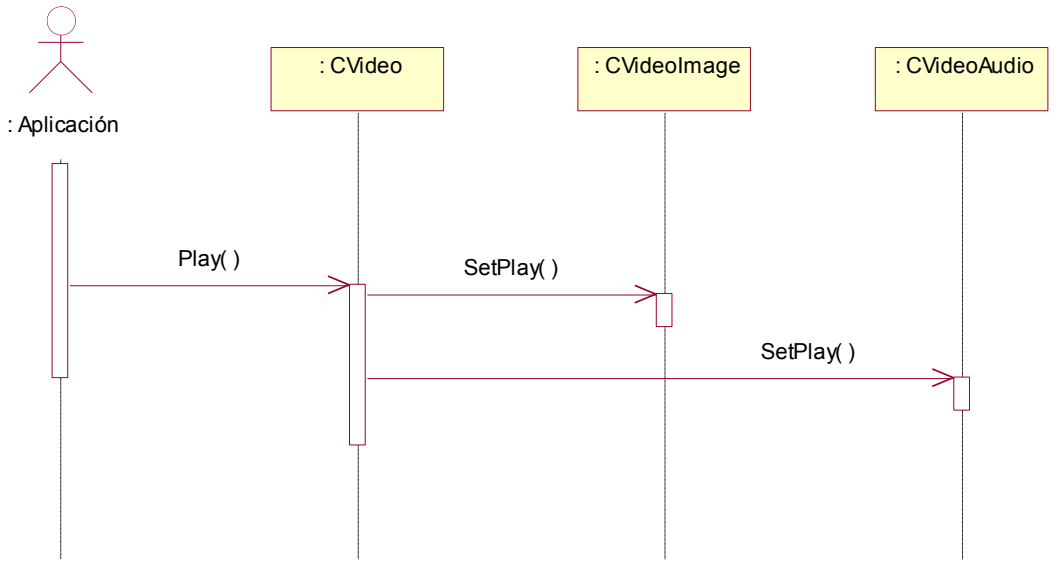

Figura 6: Diagrama de Secuencia Controlar Video Sección: Play.

El diagrama de secuencia "Controlar Video" correspondiente a la sección: Stop que se propone, en la Figura 7, muestra al actor Aplicación interactuando con la clase CVideo, la cual envía, cuando el actor desee reproducir el video, el mensaje de detener la reproducción a las clases encargadas de manipular la información referente a la imagen y al audio respectivamente. Como resultado se detiene la reproducción del video.

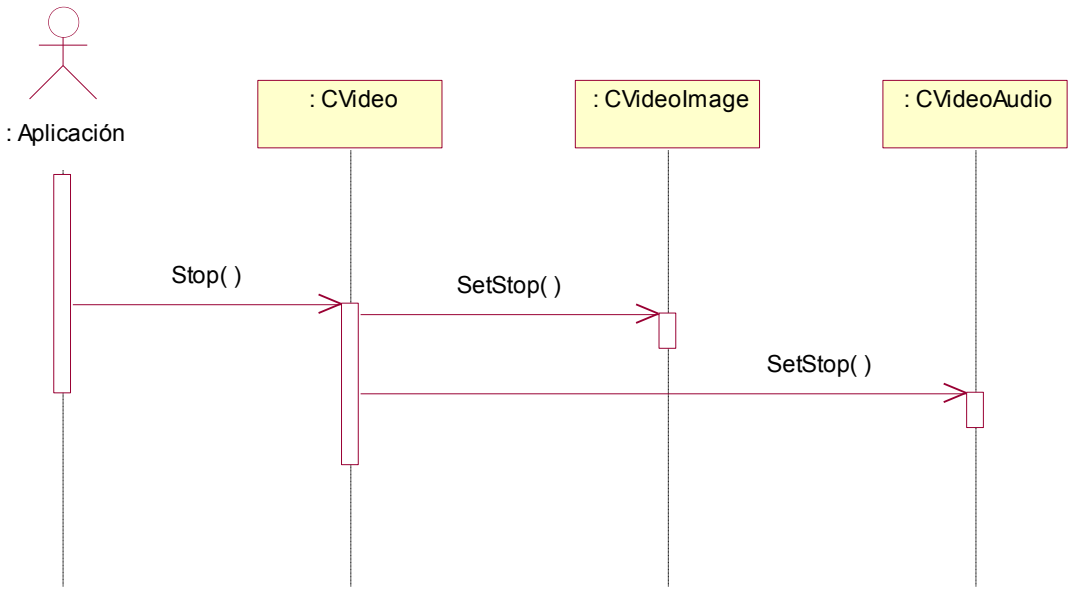

Figura 7: Diagrama de Secuencia Controlar Video Sección: Stop.

El diagrama de secuencia “Texturizar Polígono” que se propone, en la Figura 8, muestra al actor Aplicación interactuando con la clase CVideo, la cual envía a la clase encargada de manipular la información referente a la imagen del video, el mensaje de crear un polígono texturizado con la imagen que se encuentra en memoria, perteneciente a un frame del video. La clase que contrala la imagen del video actualiza la imagen en memoria dependiendo de los frames por segundo que tenga el video, luego se genera una nueva textura a partir de la imagen en memoria y por último se crean los vértices del polígono mientras se leen las coordenadas de la textura generada, obteniéndose un polígono texturizado.

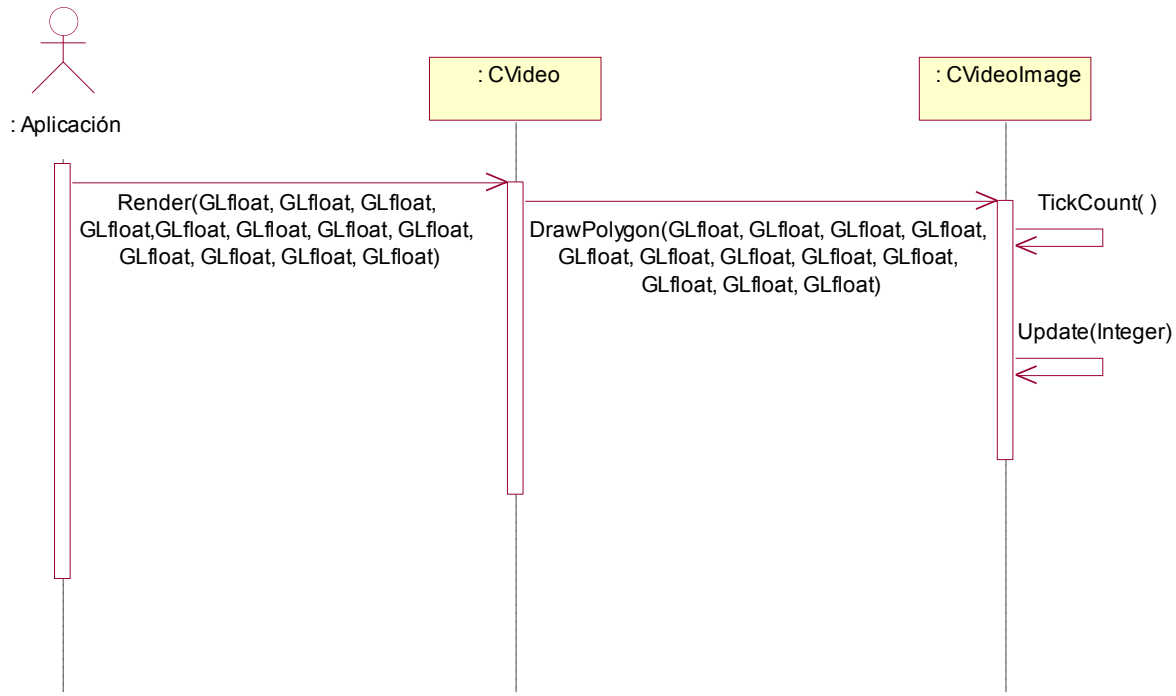

Figura 8: Diagrama de Secuencia Texturizar Polígono.

El siguiente diagrama de secuencia “Gestionar Video” que se propone, en la Figura 9, se refiere a la secuencia de mensajes necesarios para leer la información referente al fichero de video, en el diagrama se puede apreciar la diferencia que existe entre las funciones que se utilizan cuando se está sobre la plataforma Windows con respecto a la plataforma Linux. Conocer sobre que plataforma se está trabajando es algo que se logra con las líneas de código siguientes: “if defined (_WIN32)” para el caso de la plataforma Windows y “if defined (linux)” para trabajar sobre la plataforma Linux.

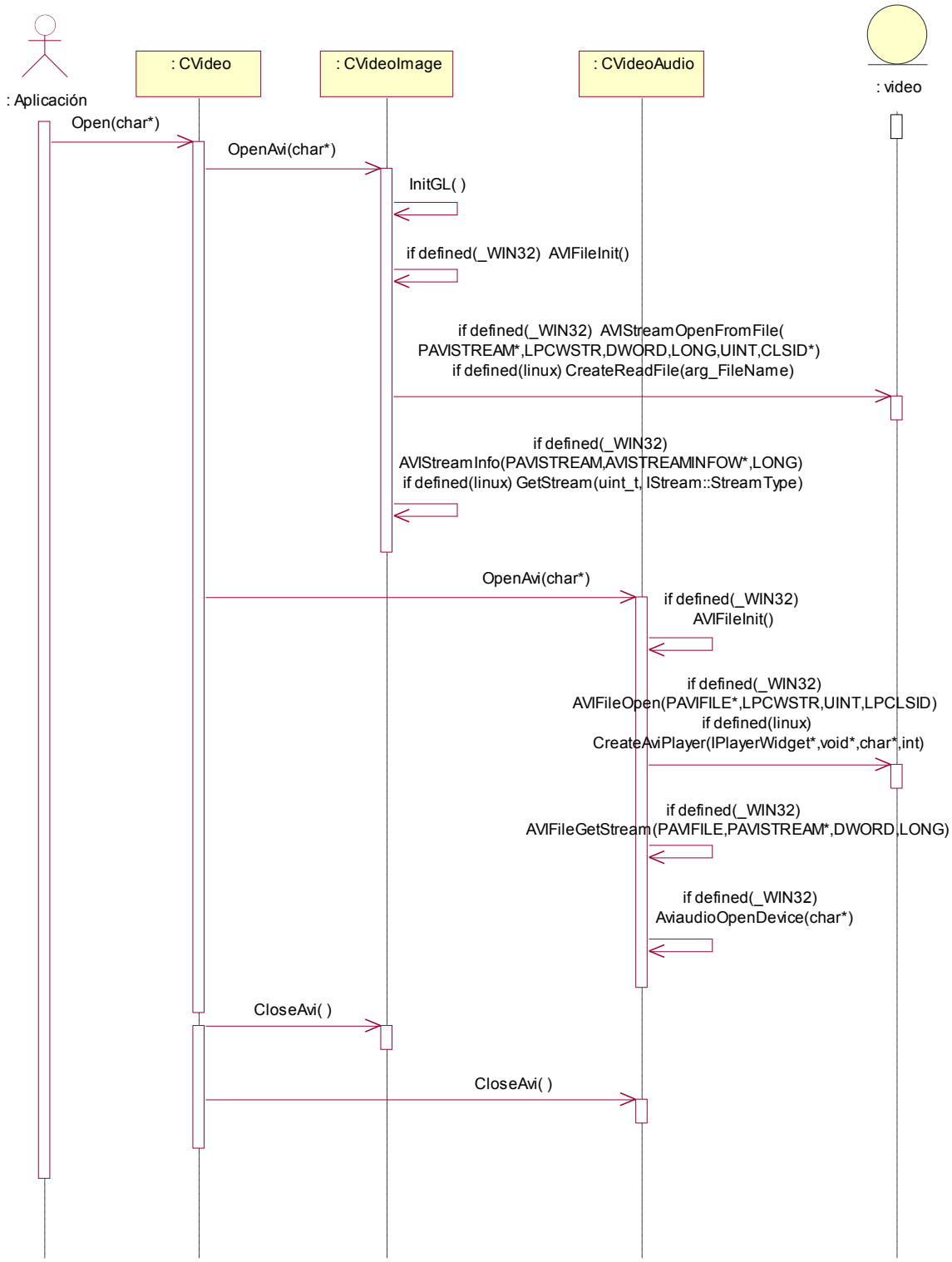

Figura 9: Diagrama de Secuencia Gestionar Video.

Por último se propone el diagrama de secuencia “Reproducir Audio” que se muestra en la Figura 10, en donde se puede apreciar que ante el mensaje de reproducir el audio del video es muy fácil la reproducción de este ya que en ambas plataformas (Windows y Linux) las funciones que se usan para leer la información referente al fichero de video proveen también funcionalidades para la reproducción del audio.

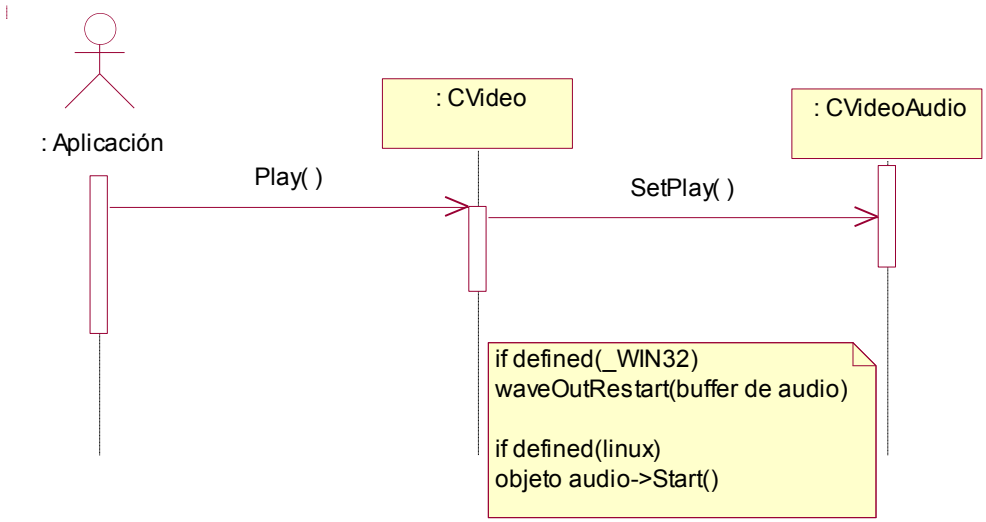

Figura 10: Diagrama de Secuencia Reproducir Audio.

3.4 Nomenclatura y Estándares de codificación.

3.4.1 Nomenclatura de las versiones.

Este módulo de video se rige por las mismas políticas para el control de versiones que la herramienta SceneToolKit. Tiene como característica que está en una fase de desarrollo primario, por lo que todavía no se ha llegado a una versión completamente estable. De esta manera, cada release nuevo que salga se considera Beta, y por demás sobrescribe completamente las versiones anteriores. Solamente se considera oficial la última versión liberada, y las anteriores se consideran como disfuncionales.

Es responsabilidad del líder del presente proyecto tener al tanto a los usuarios de la biblioteca de la liberación de la nueva versión así como la información de los cambios añadidos.

La nomenclatura de las versiones presenta la siguiente notación:

- ***.0.0**: Los cambios en la primera parte representan cambios estructurales arquitectónicos de gran impacto, así como versiones más terminadas. Constituyen las liberaciones oficiales de la herramienta, a no ser que por necesidad se requiera liberar alguna versión menos estable.

- **0.*.0**: los cambios en la segunda parte representan adición de funcionalidades o módulos de menor impacto.

- **0.0.***: los cambios en la tercera parte representan refinamiento de módulos y correcciones. Constituyen versiones internas al proyecto.
Beta: versiones para prueba por parte de los clientes.

3.4.2 Estándares de codificación.

El código de la biblioteca sigue algunos estándares propuestos por el grupo de desarrollo (respetando los estándares de codificación para C++ (identado, uso de espacios y líneas en blanco, etc.)). Está programado en inglés, debido a que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático.

El conocimiento de los estándares seguidos para el desarrollo de la misma permitirá un mayor entendimiento del código, y es una exigencia de los autores de la misma que cualquier módulo que se añada debe estar codificado siguiendo estos estándares.

Nombre de los ficheros:
Los nombres de los ficheros .h y .cpp utilizan las iniciales del nombre de la herramienta (STK).

ej: STKNameOfUnits.cpp

Constantes:
Las constantes se nombran con mayúsculas, utilizándose "_" para separar las palabras.

ej: const int MY_CONST_ZERO = 0;

Enumerados:
Para los enumerados se utiliza el indicador "E" en el nombre del tipo, y en las constantes se utilizan las iniciales del nombre del enumerado. Las constantes van en mayúsculas.

ej1: enum EMyEnum
{
ME_VALUE,
ME_OTHER_VALUE
};

ej2: enum ENodeType
{
NT_GEOMETRYNODE,
NT_GROUPNODE...
};

Estructuras:
Se utiliza el indicador "S" para indicar que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.
ej: struct SMyStruct {…};

Clases:
Se utiliza el indicador "C" para indicar que es una clase. Ver más adelante la nomenclatura de las variables miembros.
ej: class CClassName;

Interfaces:
Se utiliza el indicador "I" para indicar que es una interfaz.

ej: IMyInterfaceName

Listas e iteradores de la std:
Para los tipos de datos utilizados de la librería estándar de C++ (vector, map, multimap, etc.), se utiliza el indicador "T", con los sufijos List, Map y MultiMap según la estructura, así como el sufijo Iter para los iteradores. Además el nombre lleva el tipo de dato a almacenar en la estructura en cuestión:

ej: vector<CNode> TNodeList;
TNodeList::iterator TNodeListIter;

ej: map<> TNameMap;
TNameMap::iterator TNameMapIter;
ej: multimap<> TNameMultiMap;
TNameMultiMap::iterator TNameMultiMapIter;

Declaración de variables:

Los nombres de las variables comienzan con un identificador del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepone el identificador "m_" (en minúscula), si son globales se les antepone el identificador "g_", y en caso de ser argumentos de algún método, se les antepone el prefijo "arg_".

Tipos simples:

ej:
bool bVarName;
int iName;
unsigned int uiName;
float fName;
char cName;
char* acName; // arreglo de caracteres
char* pcName; // puntero a un char
char** aacName; // bidimensional
char** apcName; // arreglo de punteros
bool m_bMemberVarName; // variable miembro de una estructura o clase
char g_cGlobalVarName; // variable global
short sName;
void* pvName;

Instancias de tipos creados:

ej:
EMyEnumerated eName;
SMyStructure kName;
CClassName kObjectName; // objeto de una clase
CClassName* pkName; // puntero a objeto
CClassName* akName; // arreglo de objetos

CClassName* m_akName; // variable miembro de clase
IMyInterface* pIName; // puntero interfaces

Métodos:

En el caso de los métodos, se les antepone el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepone nada. Los constructores y destructores, como lo exigen los compiladores, llevan el nombre de la clase.

Constructor y destructor:

ej:CClassName (bool arg_bVarName, float& arg_fVarName);
~CClassName ();

Funciones:

ej: bool bFunction1 (…);
int* piFunction2 (…);
CClassName* pkFunction3 (…);

Procedimientos:

ej: void Procedure4 (…);

Métodos de acceso a miembros:

Los métodos de acceso a los miembros de las clases no se nombrarán “Gets” ni “Sets”, sino como los demás métodos, pero con el nombre de la variable a la que se accede y sin el prefijo “m_”:

ej: Para la siguiente variable, los métodos de acceso serían:
int m_iMyVar; //variable
int iMyVar(); // R
void MyVar(int arg_iMyVar) // W
int& iMyVar(); // R/W

3.5 Diagrama de componentes.

Se propone el diagrama de componente perteneciente a la Figura 11, para lograr una biblioteca multiplataforma. Se aprovecha la similitud en los métodos que implementa cada clase ya que lo único que varía es el comportamiento interno de las clases que manipulan el audio “CVideoAudio” e imagen “CVideoImage” del video, dependiendo de la plataforma en la que se esté codificando (Windows o Linux), por lo que se utilizan dos .cpp por cada .h en el caso de estas clases.

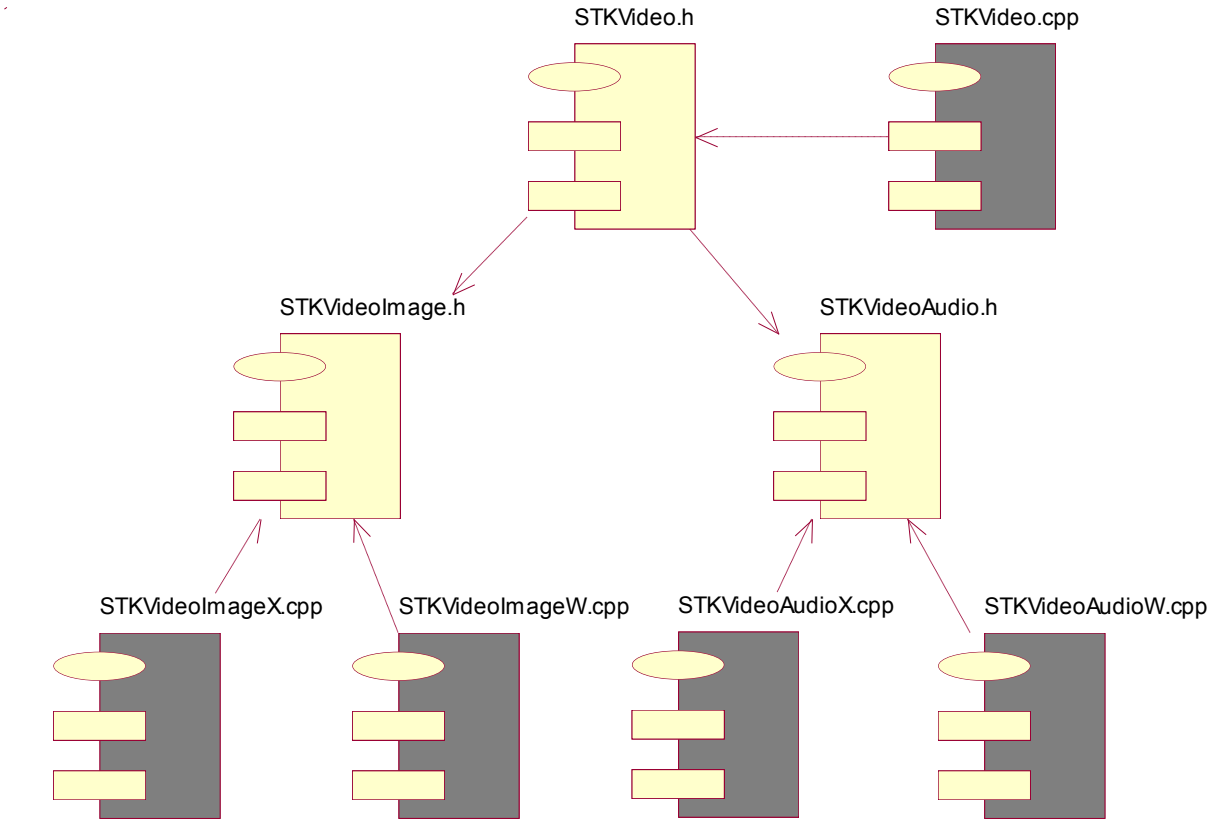

Figura 11: Diagrama de componentes.

CONCLUSIONES

Con el trabajo se obtuvo una biblioteca capaz de visualizar la imagen de un video sobre un polígono en un mundo 3D, para poder utilizar este recurso en las presentaciones de las aplicaciones gráficas que se desarrollen (ver anexo 7), utilizando la biblioteca gráfica OpenGl. Además de que permite visualizar más de un video a la vez (ver anexo 8), así como reproducir el audio de cada uno de los videos que se quieran mostrar.

Se logró una biblioteca que provee una serie de funcionalidades que abstraen al programador al nivel de sólo tener que especificar el video que se desea mostrar y las coordenadas donde lo desee mostrar. Además de poder controlar el video en todo momento.

RECOMENDACIONES

Se recomienda la integración de esta biblioteca, para la manipulación de videos digitales en Entornos de Realidad Virtual, a la herramienta en desarrollo SceneToolKit. Se recomienda que se siga desarrollando esta biblioteca para que en un futuro soporte más formatos de video.

Producto de que se logró, poder visualizar un video en un polígono localizado en cualquier posición dentro de un mundo virtual, sería beneficioso lograr también un sonido 3D, por lo que se recomienda que se emplee la biblioteca de audio OpenAl para el trabajo con el mismo.

BIBLIOGRAFÍA CONSULTADA

[1]. Molofee, Jeff (NeHe). Gamedev.net. [Online] NeHe Production. [Cited: febrero 19, 2008.] http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=35

[2]. Nix, Jonathan. Gamedev.net. [Online] gamedev.net. [Cited: febrero 21, 2008.] http://www.gamedev.net/reference/programming/features/avifile/

[3]. McGowan, John F. [Online] AVI Overview. [Cited: marzo 4, 2008.] http://www.jmcgowan.com/avi.html.

BIBLIOGRAFÍA REFERENCIADA

[1]. *Wikipedia.* [Online] Wikimedia Foundation, Inc. [Cited: febreo 10, 2008.] http://es.wikipedia.org/wiki/API_de_Windows.

[2]. *msdn.* [Online] Microsoft Corporation. [Cited: marzo 3, 2008.] http://msdn.microsoft.com/es-es/library/172wfck9(VS.80).aspx.

[3]. **Hernández, Jorge.** *monografias.com.* [Online] Monografias.com S.A. [Cited: febrero 15, 2008.] http://www.monografias.com/trabajos15/tecnologia-multimedia/tecnologia-multimedia.shtml.

[4]. **Luis.** *monografias.com.* [Online] Monografias.com S.A. [Cited: febrero 15, 2008.] http://www.monografias.com/trabajos10/vire/vire.shtml.

[5]. **Cebrián Herreros, Mariano.** *SEPiensa.* [Online] Instituto Latinoamericano de la Comunicación Educativa (ILCE). [Cited: febrero 23, 2008.] http://sepiensa.org.mx/contenidos/2005/d_video/historia/historia_2.html.

[6]. *Wikipedia.* [Online] Wikimedia Foundation, Inc. [Cited: febrero 21, 2008.] http://es.wikipedia.org/wiki/Video_%28Se%C3%B1al%29.

[7]. *Universidad Nacional de San Luis.* [Online] [Cited: febrero 27, 2008.] http://www.unsl.edu.ar/~tecno/multimedia/1.pdf.

[8]. *E.U.I.T. de Informática y Telemática.* [Online] [Cited: marzo 2, 2008.] http://lear.inforg.uniovi.es/sm/descargas/teoria/Formatos%20de%20Video.pdf.

[9]. *Wikipedia.* [Online] Wikimedia Foundation, Inc. [Cited: febrero 18, 2008.] http://es.wikipedia.org/wiki/AVI.

[10]. **Milán Tellería, Oliver W. and Corrales de la Cruz, José Rafael.** *monografias.com.* [Online] Monografias.com S.A. [Cited: febrero 12, 2008.] http://www.monografias.com/trabajos14/streaming/streaming.shtml.

[11]. **Erino, José Ignacio.** *La taberna del Grumete.* [Online] [Cited: enero 29, 2008.] http://www.eumed.net/grumetes/2004/formatoavi.htm.

[12]. [Online] Megaservice A.S.D.A S.R.L. [Cited: febrero 11, 2008.] http://www.megaservice.com.ar/Info_espe/Los%20archivos%20AVI.htm.

[13]. *Instituto Corinaldesi.* [Online] [Cited: febrero 11, 2008.] www.corinaldesi.it/Progetti/pro-storici/gioconda/ricerca/IMAGENES%20DIGITALES.doc.

[14]. *Wikipedia.* [Online] [Cited: febrero 16, 2008.] http://es.wikipedia.org/wiki/C%C3%B3dec_de_v%C3%ADdeo.

[15]. **Camacho Román, Yanoski Rogelio.** *STK 2.3.1.* [Documento .pdf] La habana: UCI, 2008.

[16]. **Kabelac, Zdenek.** [Online] [Cited: marzo 4, 2008.] http://avifile.sourceforge.net/.

[17]. **Alfaro Ferreres, Luis and Roca Estellés, María José.** *Universidad de Jaén.* [Online] Universidad de Jaén. [Cited: enero 29, 2008.] http://wwwdi.ujaen.es/asignaturas/informatica/Teoria/Software%20Grafico/Grafico-doc/Graficos%20audio%20y%20video.doc.

[18]. **Ramos Raul**, *Universidad Pompeu Fabra de Barcelona.* [Online] Universidad Pompeu Fabra de Barcelona. [Cited: febrero 22, 2008] http://www.iua.upf.es/~berenguer/recursos/ima_dig/_8_/estampas/3_1.htm

[19]. *Real.* [Online] RealNetworks. [Cited: febrero 23, 2008] http://www.realnetworks.com/products/codecs/realaudio.html

[20]. Nix, Jonathan. Gamedev.net. [Online] gamedev.net. [Cited: febrero 21, 2008.] http://www.gamedev.net/reference/programming/features/avifile/

ANEXOS

Anexo 1:

Figura 12: Need for Speed.

Anexo 2:

Figura 13: MBP

Anexo 3:

Figura 14: VirtualGT (Simulador personal de conducción).

Anexo 4:

Figura 15: Ace Combat 6.

Anexo 5:

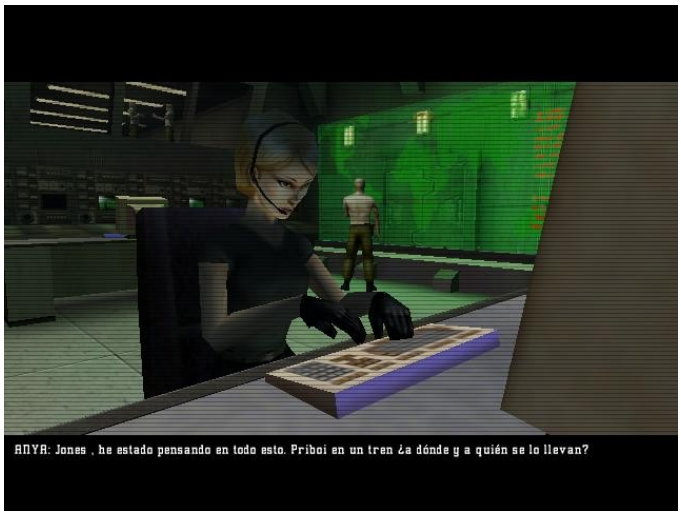

Figura 16: Proyecto IGI.

Anexo 6:

Figura 17: Fifa 2005.

Anexo 7:

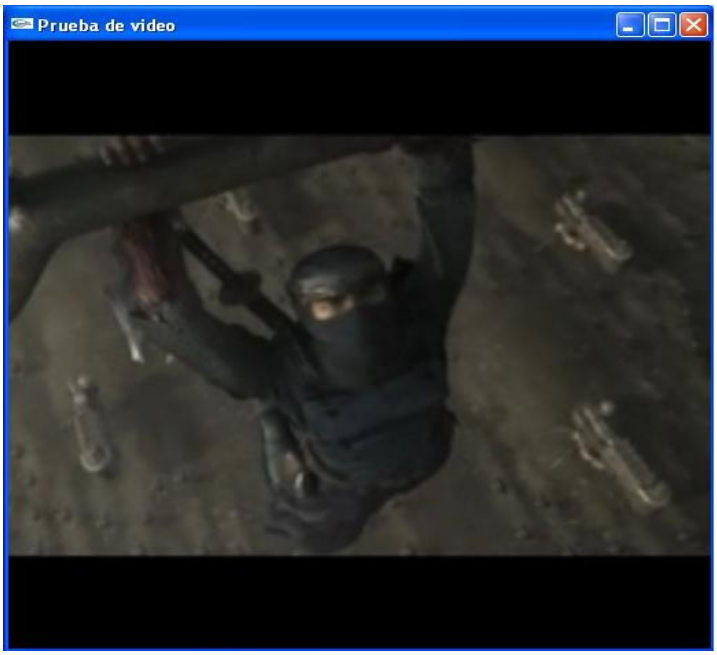

Figura 18: Utilización de un video como presentación.

Anexo 8:

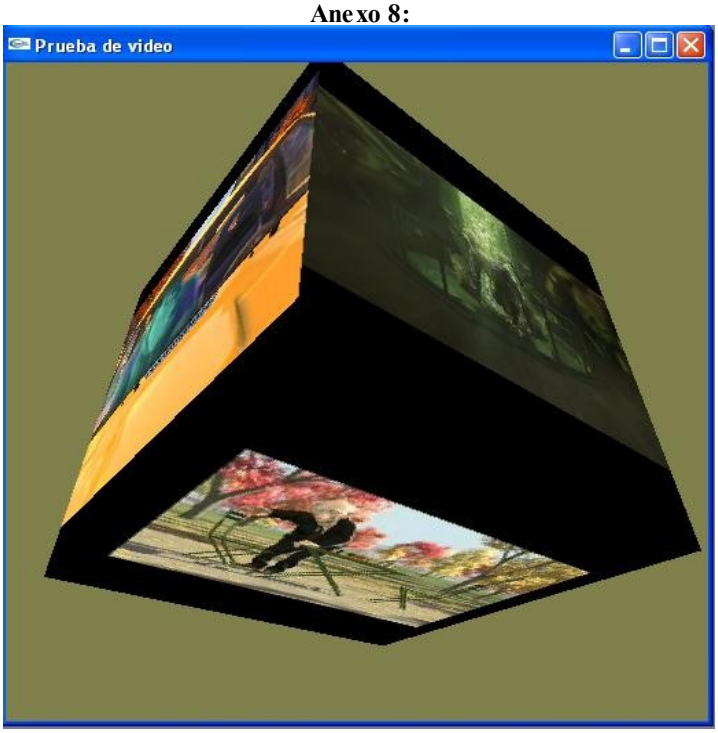

Figura 19: Cubo creado a partir de 6 planos texturizados con videos diferentes.

GLOSARIO

A:

API: Interfaz de programación de aplicaciones (Applications Programming Interface): una serie de funciones que están disponibles para realizar programas para un cierto entorno.

Archivo: Elemento en el que se guarda información: textos, datos, gráficos, fotos, películas, música. Se identifica mediante un nombre, por ejemplo, "misdatos.txt", "foto.jpg", "cancion.mp3" o "video.avi".

AVI: Es un tipo de archivo contenedor, que puede contener una pista de video codificada y una o más pistas de audio. Su nombre proviene de que el video y audio están "entrelazados" dentro del contenedor.

B:

Bitrate: Hace referencia a la velocidad con que se transmiten los datos de un contenido multimedia (video o audio). Se suele medir en Kbps (Kilobits por segundo), y por tanto indica el valor medio de Kilobits que ocupa cada segundo de contenido reproducido. Cuanto más alto sea el bitrate, más espacio ocuparán los archivos, pero también se conseguirá mejor calidad ya que hay mayor cantidad de datos disponibles y los contenidos se pueden reproducir con mayor detalle.

Buffers: Memoria intermedia, una porción reservada de la memoria, que se utiliza para almacenar datos mientras son procesados.

C:

C++: Lenguaje de programación orientado a objetos.

Campo: Cada una de las partes que forman un video entrelazado. Los campos no contienen imágenes completas, sino las líneas que forman la imagen (par o impar según el campo).

Codec: Acrónimo de codificador/decodificador. Es un software que consta de una serie de algoritmos que permiten codificar y decodificar contenidos multimedia.

D:

Digital: Representación de la información mediante combinaciones de unidades binarias o 'bits'.

E:

Entrelazado: Es un tipo de almacenamiento de video donde los fotogramas no se guardan como imágenes completas, sino en dos campos donde cada uno contiene la mitad de las líneas que forman el fotograma (un campo las líneas pares y el otro las impares).

F:

Formato: En el mundo de la informática, es el conjunto de reglas o especificaciones mediante las cuales se pueden organizar datos de diversa naturaleza, para poder acceder posteriormente a estos a través de los intérpretes adecuados.

Frame: Es cada imagen de una película, es decir, un fotograma. La sucesión de frames a gran velocidad produce el efecto de movimiento.

Framerate: Es la velocidad con que pasan los frames (fotogramas) en una película. Su velocidad se mide en fps (frames por segundo). En el estándar PAL (Europa) esta velocidad es de 25 fps, mientras que en el estándar NTSC (América) la velocidad es de 29.97 fps, aunque existe una variante NTSC Film a 23.976 fps.

I:

Interleaving: Son los puntos de conexión entre audio y video que aseguran la sincronización cuando nos desplazamos a través de nuestra película. Suponen puntos de unión a lo largo de la película entre audio y video, que el ordenador tomará como referencia.

L:

Linux: Sistema operativo desarrollado inicialmente por Linus Torvalds, un estudiante finlandés de Informática. Actualmente son miles de personas las que colaboran en todo el mundo, en su desarrollo. Se desarrolla como software libre, por lo que puede distribuirse y copiarse libremente. Existen muchas distribuciones que lo distribuyen.

M:

Multiplataforma: Que la aplicación corra en varios sistemas operativos.

Mundos virtuales: Simulación de mundos o entornos, denominados virtuales, en los que el hombre interacciona con la máquina en entornos artificiales semejantes a la vida real.

O:

OpenAL: OpenAL significa Open Audio Library, lo que en castellano significa: Biblioteca Abierta de

Audio. Pretende ser una extensión de OpenGL que provea herramientas para el manejo de audio.

OpenGL: es una biblioteca gráfica desarrollada originalmente por Silicon Graphics Incorporated (SGI). OpenGL significa Open Graphics Library, cuya traducción es biblioteca abierta de gráficos.

P:

Píxel: Es cada uno de los puntos de luz que forman una imagen en un monitor, es decir, la unidad básica que forma una imagen.

Progresivo: Es lo opuesto a entrelazado. Los fotogramas se guardan como imágenes completas y no en campos entrelazados.

R:

Realidad: cualidad o estado de ser real o verdadero.

Realidad Virtual: La Realidad Virtual es simulación por computadora, dinámica y tridimensional, con alto contenido gráfico, acústico y táctil, orientada a la visualización de situaciones y variables complejas, durante la cual el usuario ingresa, a través del uso de sofisticados dispositivos de entrada, a "mundos" que aparentan ser reales, resultando inmerso en ambientes altamente participativos, de origen artificial.

Resolución: Es el número de píxeles que se muestran en una pantalla. Al ser ésta una matriz de filas y columnas de píxeles, primero se nombra la cantidad de columnas (resolución horizontal) y luego la cantidad de filas (resolución vertical).

RGB: Es el acrónimo inglés Red, Green, Blue (Rojo, Verde, Azul). Es un modelo de color en el cual es posible representar un color mediante la mezcla de tres colores primarios: rojo, verde y azul.

S:

SceneToolKit: Motor de visualización 3D.

V:

Video digital: Una señal de video representado por computadora y puede leer números binarios que describen un conjunto finito de colores y niveles de luminosidad.

Virtual: existe o resulta en esencia o efecto pero no como forma, nombre o hecho real.

ÍNDICE DE FIGURAS Y TABLAS

Índice de figuras.

Índice de tablas.